**Pavol Bezák**

# Using Motion Planning and genetic Algorithms in Movement Optimization of industrial Robots

# Scientific Monographs in Automation and Computer Science

Edited by
Prof. Dr. Peter Husar (Ilmenau University of Technology) and
Dr. Kvetoslava Resetova (Slovak University of Technology in Bratislava)

**Vol. 5**

# USING MOTION PLANNING AND GENETIC ALGORITHMS IN MOVEMENT OPTIMIZATION OF INDUSTRIAL ROBOTS

Pavol Bezák

Universitätsverlag Ilmenau
2012

## Impressum

**Bibliographic information of the German National Library**
The German National Library lists this publication in the German national bibliography, with detailed bibliographic information on the Internet at http://dnb.d-nb.de.

Author's acknowledgement to Gabriela Chmelíková for translation.

This scientific monograph originated from the author's dissertation thesis defended at the Slovak University of Technology in Bratislava, Faculty of Materials Science and Technology in Trnava.

**Reviewers:**
Doc. Ing. Anton Kachaňák, CSc.
Prof. Ing. Juraj Špalek, PhD.
Doc. Ing. Peter Schreiber, PhD.
Ing. Augustín Gese, PhD.

**Author's contact address:**
Ing. Pavol Bezák, PhD.
Slovak University of Technology in Bratislava
Faculty of Materials Science and Technology in Trnava

Ilmenau Technical University / University Library
**Universitätsverlag Ilmenau**
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

**Production and delivery**
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

**ISSN** 2193-6439 (Print)
**ISBN** 978-3-86360-047-1 (Print)
**URN** urn:nbn:de:gbv:ilm1-2012100195

Titelfoto: photocase.com

**Abstract**

The issues of path and trajectory planning algorithms and optimization of industrial manipulator trajectory generation are still not completely solved due to their variability and increasing complexity with the growing number of robot degrees of freedom. Generation of an optimal trajectory can be solved in several ways, such as traditional numeric and more recent approaches, which include evolutionary algorithms and genetic algorithms within them.

The first chapter is devoted to a brief overview of path planning methods, especially in mobile robots. The second chapter deals with a more detailed overview of robot path planning methods in continuous and discrete environments. The third chapter describes the most popular motion planning algorithms. The fourth chapter is dedicated to genetic algorithms which we used as an optimization method. The fifth chapter focuses on optimal robot motion control and optimization methods using genetic algorithms as the method for an industrial manipulator control. The next chapter contains a solution and its implementation in support software, as well as the experimental verification of the results. The last chapter evaluates the results and their benefits.

**Key words**

trajectory planning, optimization, genetic algorithms, industrial manipulator

## LIST OF SYMBOLS AND ABBREVIATIONS

### Abbreviations

| | |
|---|---|
| GA | genetic algorithm |
| PA | probabilistic algorithm |
| RRT | Rapidly-exploring Random Tree |
| EST | Expansive-Spaces Tree |
| SBL | Single-query, Bi-directional, Lazy-collision checking |
| PRM | Probabilistic Road Map |
| C-space | configuration space |
| PMP | Pontrjagin Minimum Principle |

### Symbols

*Path and trajectory planning*

| | |
|---|---|
| $g(x)$ | distance from start to finish |
| $h(x)$ | distance from current node to finish |
| $l_x$ , $l_y$ | length of environment in x- and in y-axes |
| $n_x$ , $n_y$ | number of cells in x- and y-axes |

*Genetic algorithms*

| | |
|---|---|
| $P$ | population |
| $N$ | number of population elements |
| $S$ | population element |
| $F$ | fitness function |
| $\Theta$ | selection operator |
| $\Omega$ | set of genetic operators |
| $\Psi$ | reduction operator |
| $T$ | accomplishment criterion |

*Motion control*

| | |
|---|---|
| $q_i(t)$ | time function |
| $Q_k$ | set of tolerable system configurations |

*Optimal motion control*

| | |
|---|---|
| $u(t)$ | control |
| $x(t)$ | control response |

| | |
|---|---|
| $J(u)$ | purpose function |
| $\dot{x}$ | vector of state variables |
| $U$ | vector of control |
| $A$ | vector of non-linear state and control functions |
| $M(\varphi)$ | matrix of mass |
| $C(\varphi)$ | matrix of damping |
| $K(\varphi)$ | matrix of stiffness |
| $\varphi$ | vector degrees of freedom of general shift |
| $F(t)$ | vector of control powers driving the |
| $x(t_0)$, $x(t_f)$ | joint position at the beginning and end of motion |
| $U_i^-$ , $U_i^+$ | minimum and maximum torque generated by actuator |
| $J^*$, $x^*$, $u^*$ | optimal parameter values |
| $g_i$ | optimized criterion |
| $w_i$ | weight factor |

*Genetic planning of trajectory*

| | |
|---|---|
| $N_p$ | set of real parameters |
| $\hat{x}$ | chromosome of population |
| $x_i$ | real parameter |
| $x_i^U$ | maximum boundary value of real parameter |
| $x_i^L$ | minimum boundary value of real parameter |
| $\hat{x}_i$ | binary string |
| $\hat{L}_i$ | binary string length |
| $q_i$ | $i^{th}$ arm turn angle in momentary position |
| $\dot{q}_i$ | velocity of $i^{th}$-joint |
| $q_g$ | total angle of manipulator arms at the end of motion |
| $T_1$ | time from initial position to momentary position |
| $t_2$ | time from momentary position to destination position |
| $g_k$ | restriction of inequality |
| $h_l$ | restriction of equality |
| $M_1$, $M_2$ | restriction numbers of inequalities and equalities |
| $\phi()$, $\psi()$ | punishment functions for restriction of inequalities and equalities |

| | |
|---|---|
| $P(x)$ | punishment function |
| $Q$, $\dot{q}$, $\ddot{q}$ | generalized vectors of position, velocity and acceleration |
| $t_F$ | road time from initial to destination positions |
| $T$ | generalized vector torque control (force) |
| $M(q)$ | matrix of inertia |
| $C(q, \dot{q})$ | Coriolis and centrifugal power vector |
| $G(q)$ | vector of gravity |
| $q_{j,i}$, $v_{j,i}$, $a_{j,i}$ | position, velocity, acceleration of $j^{th}$ joint in $i^{th}$ node point |
| $fit$ | fitness population |
| $\Omega_s$, $\Omega_c$, $\Omega_m$ | operator of selection, crossing, mutation |
| $p_c$, $p_m$ | probability of crossing, mutation |
| $N_{pop}$ | size of population |
| $N_{gen}$ | number of generations |
| $\tau$, $d\tau$, $N$ | time, period of sampling, partial road time |

# INTRODUCTION

Optimization issues can occur in all fields of human activities. They emerge in such situations when it is necessary to come to some solution. Obviously, we seek the most convenient solution. Optimization issues can be handled by optimization methods. To be able to formulate the optimization matter in a mathematical way, it is necessary to constitute a mathematical model of the situation. A real situation model is always simplified, i.e. a mathematically processable situation model does not describe true reality, and vice versa, a model close to the reality does not have to be processable in a mathematical way.

The best solution selection brings along certain drawbacks. For the mathematical formulation of an optimization matter we choose such a criterion that allows us to select the best solution variable. The optimum criterion selection is problematic and in many applications it is frequently subject to subjective requirements. To solve a real optimization matter via its mathematical model, it is often necessary to specify the model and modify the optimum criterion. Various versions of the optimization matter have to be dealt with repeatedly as well as verified via simulation and comparison with reality. Transition from a real optimization matter to its mathematical model is very important and essentially influences the utilization of results.

Looking for an optimal solution via mathematical modeling is basically looking for a function extreme by which the system is mathematically described. Specifically, it is looking for local extremes which are or are not global extremes at the same time.

When elaborating a mathematical model, the increasing performance of computers should not tempt us to neglect the volume of calculations necessary to solve the optimization matter. Some systems are so complex that in the attempt to put all the essential system properties in order, we finally come to a model that cannot be used even by the most advanced information technology.

Evolutionary algorithms belong to the modern methods of system optimization. Evolutionary mechanisms verified by nature can be successfully applied to technical issues as well, mainly to complex matters and matters that are difficult to describe by mathematical methods.

Evolutionary algorithms (EA) are very efficient optimization algorithms that come out from natural genetics and its laws. Evolutionary algorithms are usually classified as **genetic algorithms**, **genetic programming** and **evolutionary strategies**.

Classic **genetic algorithms** (GA) use the operations of *selection, crossing* and *mutation* to simulate the reproduction process. Regarding the diversity of handled optimization matters, there is no generally available optimization algorithm. It is always an algorithm which is matter dependent, i.e. more or less suitable to the purpose function given. Evolutionary optimization algorithms are not suitable for applications where the purpose function gradients can be easily identified or the purpose function is difficult to calculate. The combination of non-evolutionary optimization methods (e.g. simulated annealing, the method of forbidden searching, climbing algorithms, etc.) and evolutionary optimizations are the source of hybrid algorithms. An agent with the best achieved evaluation (fitness) is considered to be the solution to the matter.

The evolution process of exploring the space of potential solutions requires looking for a compromise (balance) to achieve the following two goals:

- to find the nearest (mostly local) solution in small surroundings of the initial point as soon as possible,
- to explore the space of all possible solutions as soon as possible.

Individual methods differ according to the goal preferred.

The issue of planning the motion of a mobile robot is a frequently discussed topic. Various aspects have been researched by IT experts, engineers and mathematicians. Theoretical outcomes have led only to certain general solutions of the matter, as the matter requires an enormous amount of calculations.

Optimal motion of an industrial robot on a specific trajectory requires definition of an optimization criterion and then modification of the generated trajectory so that for example the motion performance is minimized while the maximum speed and the acceleration cannot be exceeded, so that the life of the gears is not shortened due to the significant load on joints and high moment of inertia.

Regarding the manipulation tasks, the optimization of shortening the time via the utilization of robot's available performance is becoming more and more important. Besides achieving the maximum speed, the ability of utilizing the gear units' potential to achieve an optimal acceleration in any motion point – when it is possible to shorten the cycle time in practice by 25% – is important as well. It might seem not to make a big difference; however, in large serial productions the savings are significant.

Optimization of the trajectory brings the achievement of full control over the motion and provides the space for productivity improvement with no external changes to the workplace needed, only via utilization of the robot control system possibilities.

The first chapter is devoted to a brief overview of path planning methods, especially in mobile robots. The second chapter deals with a more detailed overview of the robot path planning methods in continuous and discrete environments. The third chapter describes the most popular motion planning algorithms. The fourth chapter is dedicated to genetic algorithms which we used as optimization methods. The fifth chapter focuses on optimal robot motion control, and optimization methods using genetic algorithms as the optimization method for an industrial manipulator control. The next chapter contains a solution and its implementation in support software, as well as the experimental verification of the results. The last chapter evaluates the results and their benefits.

# 1. THEORETICAL BACKGROUND

## 1.1 Robot and surrounding environment

A robot is a mechanical device capable of performing a variety of programmed tasks. It can operate under direct human control (e.g. the robotic arm of a space shuttle) or autonomously under programmed computer control.

Robots can be divided into manipulators (industrial robots) and mobile robots. Mobile robots are capable of motion in their working environment and are not fixed to one physical place. In contrast to them, manipulators comprise a jointed arm attached to a fixed surface.

## 1.2 Degrees of freedom

Due to the variety of navigable places in the robot working environment, it is very useful to know the way to describe the position of each point of the robot in the moment fully and clearly. If the robot represents a point in a space, as is theoretically common, it can be fully described by its motion coordinates (*x, y, z)*. If the robot is a fixed solid moving freely in 3D space, six parameters are needed *(x, y, z, α, β, γ*), then the coordinate in each of three axes as well as the axis rotation to be able to describe the position of each robot's body point. Each of the parameters or coordinates is called a degree of freedom.

## 1.3 Overview of robot path planning methods

At present there are many robot path planning methods based on different principles. Each of these methods has its benefits and constraints; therefore, it always depends on the application given as well as on the complexity of its tasks.

### *1.3.1 Exact planning*

This type of algorithm is smart and efficient; however, it is applicable only to simple tasks. It does not utilize approximation and it always finds the path if there is one. If there is no path, the algorithm verifies that the path really does not exist.

### *1.3.2 Visibility graph*

A visibility graph is a graph whose nodes are represented by start and destination points and vertices of all obstacles. Only those connecting paths are selected that do not cross the obstacles. The issue of path planning is then transferred to the shortest possible path search by the graph between the start and destination points.

### *1.3.3 Retraction method*

This method uses a Voronoi diagram to search for the shortest possible path. The edges of the Voronoi diagram are represented by paths equally distant from the two nearest obstacles and its vertices are represented by points where three or more such paths meet. The search for the shortest path by the graph is the solution.

### *1.3.4 Potential field methods*

These methods use the idea of imaginary forces acting on a robot. The obstacles act on the robot by a repelling force, whereas the finish acts on a robot by an attractive force. The sum of these forces, the R result force, determines the consequent direction a speed of the path. One reason for these methods' popularity is in their simplicity and elegance. On the other hand these methods do not guarantee that the found path will be the shortest and safest one.

### *1.3.5 Dividing the space into simple areas*

One of the oldest approaches to path planning is opening the space in which the robot operates into simple areas called cells and the construction of a non-oriented graph, a so-called continuous graph. The graph represents the relation of the neighborhood among cells. A Dijkstra search algorithm or A* algorithm are usually used to generate the path.

### *1.3.6 A* algorithm*

An A* algorithm utilizes a heuristic function which, depending on the sum of distances from the start and finish of individual points, determines the order of these points. For each of the nodes we work with these three values:

*g(x)*: real distance from the start to a current node

*h(x)*: distance from a current node to the finish

*f(x)*: sum of *g(x)* and *h(x)*

In a search such a node will be selected which has the smallest value of the *f(x)* evaluation function. Since the *h(x)* function is not known, we

replace it by the *h*(x)* function which expresses the distance estimation from the given node to the destination. It is called a heuristic function and it is essential for search efficiency. The path made by the start point is placed as the first one to the front of paths. Then the paths are taken from the front until the last point is the same as the destination; then the path is the solution. Otherwise, new paths are made by joining this path and adjacent points. These new paths are filed into the front in the order according to the distance from the finish. The points once crossed by the algorithm are filed into the file of closed points, and the paths ending by such a point are not further processed.

### *1.3.7 Probabilistic planning*

This type – as the name suggests – is based on probability. The way of planning can manage a non-convex robot (a turning robot), robots with limited motion (a car), and motion dynamics represented by inertia as well as limited acceleration. To describe the probabilistic planning it is necessary to define the configuration space indicated as C-space. C-space is *n*-dimensional space, where *n* is the number of parameters unambiguously defining the robot position or configuration. That means that instead of being interested in the environment in which the robot operates, we are more interested in the number of parameters describing its configuration.

### *1.3.8 Genetic algorithm - GA*

This method was discovered in the 70s of the last century and is based on the application of Darwin's theory of natural selection for the solution to complex situations where classical mathematical and physical approaches fail. I will deal with this method in more detail in Chapter 3.

# 2. MOTION TRAJECTORY PLANNING

## 2.1 Global and local planning

Global planning is aimed at finding a non-collision path from start to destination configurations. Global planning is made before the robot makes the first motion and requires that the environment is completely known, i.e. the path map is at disposal, including all the static obstacles to avoid. This path is then sent for further processing to the local planner which controls the robot and considers possible constraints (robot motion constraints, new obstacles, etc.) that occur in the course of going on the path. The task of the local planning is the control of the robot on the path planned within the global planning.

## 2.2 Holonomic and non-holonomic planning

Regarding the constraints on the robot's motion we distinguish two basic kinds of motion planning – holonomic and non-holonomic planning.

In classical mechanics a system can be defined as a holonomic one, if all its constraints are holonomic. Holonomic constraints are such constraints which can be expressed as functions $f(x_1, x_2, x_3,...x_n , t) = 0$, i.e. constraints depend only on coordinates of the system and time. The constraint does not depend on the velocity or mobility of the system.

In robotics holonomity expresses the relation among the number of the robot's controllable degrees of freedom and their total number.

If the number of controllable degrees of freedom is the same as their total number, then we can say that the robot is holonomic.

If the number of controllable degrees of freedom is smaller than their total number, then the robot is non-holonomic.

A car is an example of a non-holonomic system, as its motion to the sides is limited by the maximum turn of its front wheels.

## 2.3 Complete motion planning algorithms

An algorithm for motion planning is complete if the search between two robots' configurations is guaranteed, if there is a path; in the opposite case it announces that there is no path. Complete algorithms are sometimes indicated as exact algorithms.

## 2.4 Path planning

In recent decades the matter of roadmap planning is frequently discussed by the scientific community. Within the basic matter one robot is in the static and known environment, and the task is to calculate a non-collision path describing the motion that replaces the robot from its current position to some desired position. There many varieties of the matter.

In general, the solution to even the basic matter of the path planning requires time growing exponentially with the number of degrees of freedom. Many so-called complete planners have been developed that are not applicable in various practical situations, because they find no solution. Many researchers have tried to make the path planning complexity more simple.

Completeness is a preferred property of motion planners, and probabilistic completeness in particular. The planner is probabilistically complete if in the solution to the matter the probability approximates a value of 1 as the running time approaches the infinity.

The environment in which the robot moves is static (does not change in time). In its search dynamic changes do not occur. The environment can be plastic with rises and falls. It can simulate a certain state.

### *2.4.1 Discrete environment*

A right-angled chessboard-like network consisting of cells is the basis of this environment. A space originated from three cells is called a scene (2D). This scene is of a rectangular shape.

### *2.4.2 Continuous environment*

A continuous environment is not divided into a square network as known by the discrete environment. It is a continuous space that can be compared to the environment around us. The robot can move in a random direction.

The obstacles are surface unevenness, objects on the ground, etc., in fact the environment we move in. These obstacles limit the robot's mobility. To simplify that, we mainly consider one type of obstacle – non-transferable static. There are also obstacles that can be overcome. The robot, in overcoming such obstacles, has to make specific operations, e.g. crossing over, etc.

Another classification of obstacles is static and dynamic ones. Dynamic obstacles are, for example, people or other robots. Static obstacles do not change their position or size during the robotic motion.

The representation of obstacles is qualified mainly by the type of environment. For instance, in a discrete environment the obstacle comprises one or more inseparable units. In a continuous environment the obstacle is usually defined by its vertices and edges.

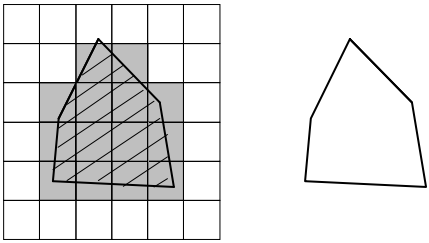

***Fig. 1*** *Representation of obstacles.*
*In discrete (left) and continuous (right) environments.*

### *2.4.3 Robot movement*

Robots move in continuous or discrete environments according to 2D or 3D dimensions in space. We choose two random points of the space as the start and destination robot positions (there should be no obstacle in these places). The task is to find the path which the robot can take to go from the initial to destination positions. The path should not cross any obstacle.

The robot motion speed is considered as a constant for the calculation simplification or as a variable.

From the point of view of the searching algorithms, the robot motion possibilities are limited in the discrete environment by the way of discretion. For instance, in a chessboard discrete environment the robot can move in eight directions, as illustrated in Fig. 2. Four directions are in the

main axes directions (up, down, left, and right) and four are in the directions of diagonals (left up, right up, left down, right down).

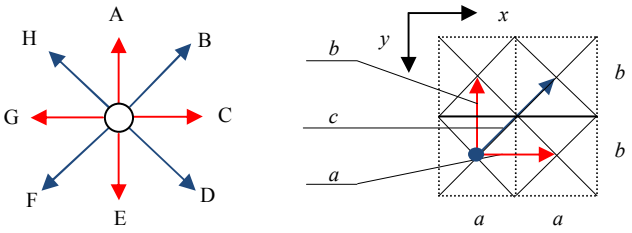


***Fig. 2*** *Chessboard environment (a) directions of motion; (b) distances of motion*

In the continuous environment the robot is not limited in its motion by the way of landscape representation. In contrast to discretion landscape it is usually specified by more complex difficulty of searching.

### *2.4.4 Probabilistic algorithms*

Probabilistic algorithms (PAs) work on the basis of random sampling of the continuous environment and their subsequent connection into a graph by some basic deterministic algorithm. The calculation of the path exploration is played out before the robot motion; therefore the probabilistic algorithms are sometimes indicated as offline algorithms.

### *2.4.5 Use of probabilistic algorithms*

Robotic arms in production lines are a typical utilization of probabilistic algorithms in 3D environment/space. They frequently have a large number of degrees of freedom – in that case every degree of freedom

is represented as another environment/space dimension; therefore PAs are very suitable for such tasks.

In a 2D environment PA either non-holonomic (e.g. robot type – a car) or holonomic (all directional) robots are utilized for the robot motion control. It is also possible to include the motion dynamics by using the so-called algorithms based on the control.

### *2.4.6 Classification of planning algorithms*

Basic classification of the probabilistic algorithms is based on the fact of whether the graph built by them and the recording of the space given can be used repeatedly – i.e. for exploring the way among various points:

- A single query-algorithm builds a graph between two specific points in space and the graph is not usable repeatedly for exploration between two other points. RRT an EST belong here.
- A multiple query-algorithm first builds a graph (a network of points, road map) recording the space given and via this graph it is possible to explore the path between two random space points repeatedly. PRM and its modifications belong here.
- A combined query – SRT algorithm belongs to this category, and it stands on the border of the aforementioned queries as it is possible to be utilized for repeated exploration between two random space points; however, they can be faster than the single query and what is more, it utilizes the function of the single query internally by itself. In general, SRT is the most efficient algorithm.

There are differences among the specific algorithms and it is also important which space they operate in, nevertheless, we can state that single query-algorithms are faster for finding the path between two specific space points, however, if we want to explore the space repeatedly, we had better utilize multiple query-algorithms which take a longer time to build the graph, but they are faster in repeated path exploration than their single query counterparts (1).

# 3. GENETIC ALGORITHMS

A genetic algorithm (GA) is a heuristic approach trying to look for the solution to complex issues by the application of the principles of evolution biology, if there is no exact algorithm available. Genetic algorithms or all procedures classified as the so-called evolution algorithms use techniques simulating the evolution processes known from biology – heredity, mutation, natural selection and crossover – for the "improvement" of the solution to the task given.

The principle of the genetic algorithm lies in the gradual building of the generations of various solutions to the issue. In the solution a population in which each individual represents one solution to the issue given is kept. As the population undergoes evolution, the solutions improve. Traditionally, the solution is represented by binary numbers, strings of nulls, and units, however also other representations are used (tree, field, matrix, etc). At the beginning the simulation (in the first generation) population is typically composed of completely random individuals. In the transition to the new generation, the so-called fitness function expressing the quality represented by the member in question is calculated for each individual. Due to this quality the individuals are selected at random, then modified (via mutation and crossover), which leads to the origin of a new population. The procedure is repeated iteratively, which makes the solution quality gradually improve. The algorithm is usually stopped after achieving a sufficient solution quality, or possibly in the period given.

### 3.1 GA definition

A genetic algorithm is a random adaptive algorithm comprising the following operators and parameters:

$$GA = (N, P, f, \Theta, \Omega, \Psi, \tau)$$

where $P$ is the population of $N$ elements (individuals), $P = \{S_1, S_2, ..., S_N\}$. Each element $S_i$, $i=1,..., N$ is a string (or a set) of whole numbers of the fixed length n, representing the solution to the issue, i.e. $S_i \in Z^n$.

$f$ indicates the so-called fitness function, which assigns each of the elements a positive real number:

$$f = S_i \rightarrow R+;\ i = 1,\ldots,N$$

$\Theta$ is a selection operator of parent elements – a parent selection operator which selects $u$ elements of $P$:

$$\Theta : P \rightarrow \{P_1, ..., P_u\}$$

$\Omega$ is a set of genetic operators, including crossover operator $\Omega_c$, mutation operator $\Omega_m$ and possibly other specific operators which altogether generate v offspring, the children of $u$ parents:

$$\Omega = \{\Omega_c, \Omega_m, \ldots\} : \{P_1,\ldots,P_u\} \rightarrow \{O_1,\ldots,O_v\}$$

$\Psi$ is the deletion operator deleting $v$ selected elements in the current population $P$. Then $v$ offspring are added to the new population $P(t+1)$:

$$P(t + 1) = P(t) - \Psi(P(t)) + \{O_1, ..., O_v\}$$

$\tau$ is the criterion of the end:

$$\tau : P(t) \rightarrow \{true,false\}$$

the parent selection operator $\Theta$ and genetic operators $\Omega$ are of probabilistic character, whereas the deletion operator $\Psi$ can be deterministic.

### 3.2 Size of population

By selecting the size of population $N$, we have considered two contradictive requirements:

- variety
- rate of convergence.

It is obvious that in selecting a small population there is also a small initial variety of elements in the population, and therefore the population tends to converge fast, however most frequently to the local optimum instead of the global optimum. In the opposite case, in the selection of a large population, there is a large initial variety of elements in the population, which means that GA has a bigger chance to find the optimal solution. Obviously, the price we pay here is the lower convergence and an increased number of algorithm operations. The size of the population in usual scope of $50 \leq N \leq 200$ fully meets the majority of issues.

### 3.3 Initial population

The initial population is either generated at random or is achieved as a set of a “good solution” via another heuristic method or from the previous genetic algorithm calculation.

## 3.4 Chromosome representation

The solution of the combinatory issue can be represented by the final set of parameters or variables acquiring discrete values. These parameters (in GA indicated as genes) make strings of values (chromosomes). In classic Holland GA the chromosome is represented by the string of binary values. Nevertheless, it is not the only way.

## 3.5 Fitness

The value of the fitness function determines the rate of the chance of the individuals in the population for the reproduction and survival to the next generation. The simplest definition of the fitness function is the direct use of the purpose function of the issue solved. In GA elements with the highest value of the purpose function by the maximization matters are preferred. By the minimization matters it is necessary to modify the fitness function, e.g. we subtract the purpose function from a specific invariable $f_{max}$, which is higher than all the values of the $f_{max}$, has to be selected sufficiently high, and then the values $f_{max} - f(Si)$ *for* $i = 1, ..., N$ can be quite close which eliminates the differences between "good" and "bad" solutions and makes the selection of elements for further operations complicated.

## 3.6 Selection of parents

The mechanism of parent selection plays a key role in GA if we want to select $u$ parent elements, and then it seems that the best thing is to select

the individuals whose fitness function values are in the first *u* places of the not growing sequence of the values. Unfortunately, this strategy results in lower genotype diversity and the individuals in the populations of the following generations are gradually concentrated only in one part of the exploring space. This can mean that the procedure can converge to the local extreme. With certain exaggeration we can say, that it comes to the similar effect as by the offspring degeneration, whose forefathers are in close family relationships. To avoid these unwanted effects for the parent selection, strategies based on probabilistic rules are used. Tournament, roulette and ordered selection are the most used strategies.

### 3.7 Genetic crossover operator

A crossover operator is generally considered to be the most important exploration operator. The crossover operator combines the segments of selected parent elements. The aim is to build new elements leading to better solutions. We mostly use discrete, point, diagonal and average crossover.

### 3.8 Genetic mutation operator

Mutation does not infer the appearance of the new generation very much; nevertheless, it has an important function. By the mutation it comes to a random change at a random place in the chromosome. At first sight it may seem that the mutation in the overall mechanism is useless, however, as already mentioned, it has important functions. Due to this marginal change, the sufficient variety of the whole population is ensured. It can

come to the increase of the fitness function with the chromosome given, since by the mutation a better solution has been achieved. The mutation hinders the situation when a lot of individuals undergo such a crossover that a further crossover would produce the same individuals and would be useless. In practice it means that it is an attempt to find solutions also beyond the original area. We utilize single point or multiple point mutations.

### 3.9 Replacement scheme

The change of a population is a replacement scheme. Immediately, as the *v* offspring are generated, these offspring replace *v* elements in the current population (the size of population stays invariable) and the reproduction cycle is repeated. By the change of population the generation exchange and incremental replacement are used.

### 3.10 Criterion of ending

Regarding the fact that we do not know the optimal solution for the practical tasks of a large scope, the ending of GA is controlled by the achievement of a defined value of a specific parameter. In analogy to some iterative methods of the numerical mathematics, two basic strategies are mostly used:

- maximum number of generations $t_{max}$ ,
- intergenerational relative improvement of the fitness function value of the best population solution.

This cycle is repeated as long as the sufficient solution is found. By the strategy of the selection of specific genetic operator types, the achieved results change significantly, therefore it is necessary to try deploying several types and then evaluating their successfulness for the solution to the specific matter.

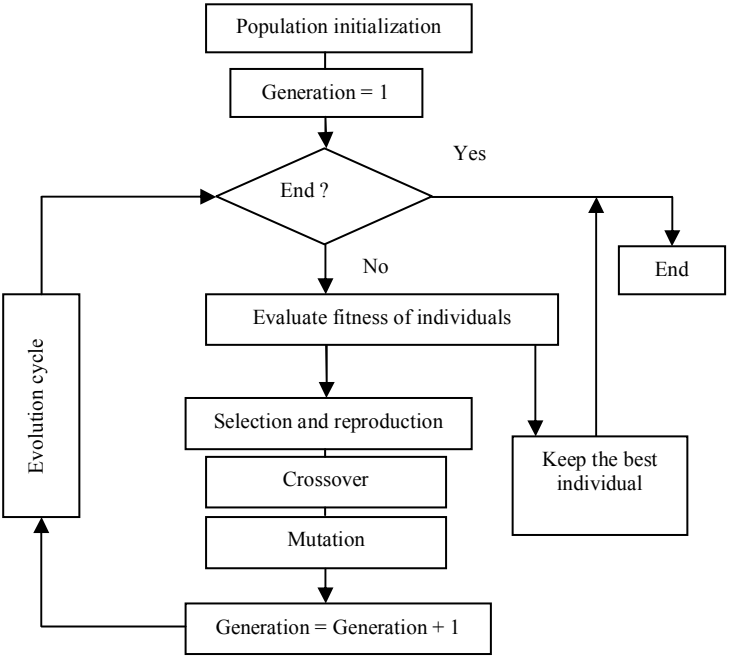


***Fig. 3*** *Flow chart of a genetic algorithm*

# 4. ROBOT MOTION CONTROL

Non-linear algorithms of the manipulator motion control are the basis of the motion planning and they utilize the solution of the direct and feedback tasks on the position of the executive robot mechanism.

The second task is to plan the type elementary motions of the executive mechanism building the basis of technological operations. A random complex trajectory can be composed of type elementary motions.

The direct task on the mechanism position determines the position and orientation of the gripper regarding the knowledge of the mutual motions of the kinetic scheme individual members. The task is solved via the relation that can help determine the coordinates of the robot's destination point in the system of coordinates connected to the base.

By the solution of the feedback task the generalized variables at the known vector specifying the position of the robot's end link are determined. In this case it is necessary to deal with the system of non-linear algebraic equations arising from the relation for the direct kinematic task. Regarding that, we need six parameters to determine the position; the initial set usually comprises six unknowns. Six degrees of freedom is also an essential prerequisite for achieving the required point with the required gripper orientation. For the systems with more degrees of freedom it is suitable to introduce additional conditions.

Information on the link position and orientation is first known for the end link. It is necessary to determine the characteristic link positions and orientation for the largest possible amount of kinematic scheme members.

The vectors characterizing the position of the end link in the coordinate systems of links are calculated gradually for the links in the direction from the end link. The result of the feedback task is a graph which each branch corresponding to a certain configuration of the kinematic scheme.

Planning of the trajectory is also required for the simplest motions. On the path of the motion we determine several node points, for which it is necessary to define the joint variables by the solution to the feedback task. Therefore, it is possible to make up a table of points from relevant joint variables. On the basis of this table, for the known way of interpolation and known boundaries it is possible to state the feasibility of the researched variation of replacement. If it is not possible to execute the motion desired, then it is necessary to utilize the in-definitiveness of the feedback task solution, to select another set of joint variables and to repeat the procedure. If it is not possible to meet the requirements for boundaries for any of the sets of variables, then we have to change the trajectory, or possibly use places for laying the object aside and to grasp it then again. In the regime of the transition, the force is adjusted to the prescribed value, while in specific cases we can use the sensors of object slipping in the grab.

Making up the table of variables is not sufficient for the robot servo systems control. For this activity time functions $q_i(t)$ are defined, where $i$ is the number of a kinematic couple. In the solution it is necessary to determine the decomposition of the whole trajectory into sections and the way of interpolation in these sections.

The planning of the trajectory is carried out by the operator, or it is carried out on the higher control level. On the basis of the known trajectory and by the solution of feedback tasks the control orders for the drives of

individual degrees of freedom are determined. Regarding the fact that this is the solution of non-linear equations, the solution is frequently executed via computers or via physical models.

Approximate solution can be achieved by the linearization of the equations describing the kinematic scheme. This description is due to small changes of coordinates.

Linearized equations for velocity increase in links' position and working forces represent the basis for the solution to tasks for the manipulator drive control.

The operator determines the desired motion speed of the end link, replacement of the robot's work parts, or possibly the force activity on the end link. Regarding this information we have to determine the drive's activity. In the process, the following methods of planning the motion trajectory of the kinematic scheme are possible: control according to the speed vector, control according to the position increase, and control according to the force vector.

Control according to the force vector is characterized by assigning the motion speed by the projections of the angle force vector speed of the work part in the coordinate system which the control system cooperates with, so that the motion velocity of the end link in the given trajectory point is determined.

Desired values for servo systems can be then set as an integral of generalized speeds. In the solution, very high desired values can occur and they cannot be precisely monitored by the servo systems. Therefore, the algorithms should be modified in order to accept only feasible solutions, or solutions ensuring a minimum error in the desired motion execution.

The components of the speed vector of the end link are set to the operators or they are generated automatically. In the first case the execution of desired trajectories are checked in the presentation mode to the operators, because the system is positionally closed through the operator. The specific motion speed of the end link corresponds with each of the positions of the given set of signals. Such a way is effective in the necessity of fast transition of the end link from one position to another and if high accuracy of the position is not required. In the other case, the operator sets the change of the end link position and the system determines the way to achieve the desired position. The speed vector is generated on the basis of regulation deviation of the end link position from the desired position.

In the control synthesis according to the speed vector, approximate solutions coming out of the boundaries of the given coordinates values are used. We usually consider three sets of coordinate values (two marginal and one in the middle) and the inversion matrix is quantified for them. For all other coordinates the inversion matrix is determined by the interpolation. An error occurred by the interpolation is usually negligible.

The method of gradual position correction is used in numerical control systems. Then the control algorithm according to the speed vector is specified as an increase of coordinates in one algorithm calculation cycle.

In the implementation of the aforementioned method it is necessary to select node points on the complex trajectory sufficiently close to each other, so that the transition from one point to another ensures the desired trajectory shape.

The control process and related calculations are simplified if the force vector control is used. The robot's servo systems develop such generalized

forces which are dynamically equivalent to the forces given replacing thus their activity on the kinematic scheme. The control system calculates the generalized forces for the coordinates controlled by the drives from the setting signals.

Redundant degrees of freedom and existence of borders/boundaries of generalized coordinates do not permit investigation of the linear equations describing the system by standard methods. Nevertheless, for the given manipulation system configuration it is possible to write all the boundaries as linear equations and inequalities via linear programming method. The control of the trajectory motion planning is based on the use of a linear model and has the following stages:

- determination of current values of generalized coordinates of q manipulation system elements and the control target determining the destination position of the robot end link,
- calculation of the continuous value of the end link position and generation of the control vector of this position change if the target has not been achieved yet,
- construction of a linear model, calculation of the transition matrix and boundaries dependent on the continuous manipulator and system state configurations,
- determination of generalized coordinates q growths via the solution to the task of linear programming,
- delivery of control signals q to the executive level and return to the first point.

If the degrees of freedom is insufficient for the given motion execution, then only the probable solution with the error minimization is determined.

The method of dynamic programming requires a precise solution to the feedback task in the node points and is suitable for kinematic schemes in which the feedback task can be solved only analytically. The difficulty of the solution is in the fact that a certain point of the space in the systems with higher number of degrees of freedom can be achieved by various combinations of the joint variables. In contrast to the feedback task, the position of the final/destination point is clearly determined by the joint variables assignment. By this method, at the beginning the feedback task for the given sequence {rk} k = 1,...N of destination link positions in the work space is solved. This results in the sequence Qk of the set of permitted system configurations. Such a configuration is permitted for which the values of generalized coordinates correspond with the construction boundaries to the scope of their changes:

$$q_{k\,min} \leq q_k \leq q_{k\,max}\,,\ k=1,...N$$

If some values of generalized coordinates cannot be determined, they are lain as equal to the values in the previous node point Qk-1. The trajectory will be made by the sequence of transitions from one node configuration to the other one and the task leads to the selection of the optimality indicator (power, time, etc.).

Industrial robots represent complex mechatronic devices comprising more functional subsystems which have to ensure various types of robot activities.

The majority of industrial robots which are currently used in practice are industrial robots of a stationary type. They represent such robot types which are firmly anchored to the base and their change of the manipulation space is possible only on the basis of the kinematic structure pre-configuration. Meeting the requirements for technically and economically effective robot implementation is possible mainly on the basis of a modular approach to robotic devices.

Mechanical systems with more degrees of freedom made mainly of open kinematic chains are the basis of industrial robot construction. By the mechanical robot concept, besides the degrees of freedom it is necessary to consider also the kinematic principle ensuring them. By the kinematic solution of the robots' mechanical systems the matter of the robot's working motions by the executive link defined motion is also implicitly determined. By the prescribed effector position in dependence on time, it is possible to define the kinematic functions of the track control, which is called the inverse task of kinematic robots. After the determination of kinematic control functions, it is possible to derive the dynamic functions of drive controls as well (3).

## 4.1 Optimal robot control

Since industrial robots and manipulators are determined to repeat predefined tasks at a high number of repetitions, even small improvements of their performance can lead to valuable time, power, or financial savings.

In this contribution I try to minimize the time of the motion action between two points regarding the best possible utilization of available robot servo gears.

## 4.2 Formulation of the robot optimal control matter

Manipulator dynamics connect the control u(t) to the dynamic response x(t). Typically, there are two kinds of matters where the dynamics utilize the manipulator's proposal. The first one is the issue of inverse dynamics, when the trajectory x(t) is known and control forces have to be determined. The other issue of the direct dynamics is when it is necessary to determine the behavior of the manipulator for certain forces.

The aim of the optimal control playing a significant role in the proposal of advanced systems is to determine simultaneously u(t) and x(t), which could be minimized by a certain criterion - functional. In the optimal control the functional of quality of the dynamic system (hereafter the functional of quality) is expressed as follows:

$$J(u) = \int_{t_0}^{t_f} g(x,u,t)\,dt \rightarrow \min \qquad [4.1]$$

It is expected that at least one *x(t)* and one *u(t)* exist, and they meet the conditions. Such a solution is considered as optimal.

Optimal control requires a mathematical process model, which is to be controlled, and then it needs the determination of physical restriction and quality assessment.

### *4.2.1 Dynamics of manipulator*

In control theory the system state I am dealing with here, and which represents the mathematical model of the manipulator dynamics, is usually written as follows:

$$\dot{x} = a(x, u, t) \qquad [4.2]$$

where $x$ is the vector of state variables, $u$ is the vector of control, $a$ is the vector of non-linear functions of states and control. In the structural dynamics the motion equations are usually written as follows:

$$M(\varphi)\ddot{\varphi}(t) + C(\varphi)\dot{\varphi} + K(\varphi)\varphi(t) = F(t) \qquad [4.3]$$

where $M(\varphi)$, $C(\varphi)$ and $K(\varphi)$ are non-linear matrices of weight, damping and stiffness. $\varphi$ is the vector of the degrees of freedom of the general shift and $F(t)$ is the vector of control forces driving the system. The equations [4.3] can be simply rewritten as [4.2].

Each component $\varphi_i$ in the vector of degrees of freedom represents two state variables: $x_k^d = \varphi_i$ a $x_l^v = \dot{\varphi}_i$, where $k = 2i-1$, $l=2i = k+1$. $i=1,..., n$. The upper indexes $^d$ and $^v$ represent the shift and velocity of shift change. This allows the division of the state vector $x$ into two parts:

$x = [x_1^d, x_2^v, x_3^d, x_4^v, ..., x_{2n-1}^d, x_{2n}^v]^T$. In the substitution into (4.3), motion equations the perception of state variables are as follows:

$$\dot{x}_k^d = x_l^v$$

$$\dot{x}_l^v = M_{ij}^{-1}[F_j - (C_{ji}x_l^v + K_{ji}x_k^d)] \qquad [4.4]$$

where $i, j = 1, \ldots, n$. This means a random system with $n$ degrees of freedom can be defined by $2n$ state variables.

### *4.2.2 Marginal conditions and restrictions*

A mathematical model of the discussed matter should also comprise the marginal conditions and physical restrictions given by the states or control elements. The marginal manipulator conditions are as follows:

$$x(t_0) = x_0 \quad x(t_f) = x_f \qquad [4.5]$$

where $x(t_0)$ and $x(t_f)$ represent the positions and velocities of joints at the beginning and end of the motion. There are also state restrictions/limitations as follows:

$$x_{min} \leq x(t) \leq x_{max} \qquad [4.6]$$

Any state trajectory meeting these state restrictions during the whole motion is called the permitted trajectory. The restrictions of control elements are:

$$U_i^- \leq u_i(t) \leq U_i^+ \qquad [4.7]$$

where $U_i^-$ and $U_i^+$ are minimum and maximum forces or moments, which can be generated by related drive engines. If the history of control instructions meets the restrictions of control elements during the whole motion, we can talk about permitted control.

### *4.2.3 Functional of quality of dynamic system*

In optimal control the functional of quality of the dynamic system is minimized or maximized. The designer of the optimized system should take several quality assessments into consideration before s/he selects the specific optimization target.

For instance, large structures or manipulators used in outer space applications are manufactured flexibly due to the requirement of a high cost decrease for material transferred to orbit. Nevertheless, higher flexibility

can introduce additional vibrations influencing the maneuver accuracy. Outer space manipulators could be optimized with respect to their accuracy as well as to their weight. This can be achieved by vector optimization, which can include these two aims.

If we want to control the manipulator by a given general task within the individual limits of control elements ($U^+$, $U^-$) and space ($x_0$, $x_f$), we can use the theory of optimal control, while the functional of quality

$$J(u) = \int_{t_0}^{t_f} g(x,u,t)\,dt \qquad [4.8]$$

has to be minimized.

Note that formally the left side of (4.8) should be written as $J(x, u)$; however, as $u$ and $x$ are connected by a state equation, the performance depends only on the control. The achievement of such an optimal control depends on the specific formulation of $g(x,u,t)$. If $g=g_1(x)$, the corresponding functional can be used to suppress the vibrations or to monitor the specific path. For instance, for $g=g_1(x)=x^TKx$, where $K$ is the matrix of toughness, the performance represents the deformation system power. If $g=g_2(u)$ the corresponding functional can be used for fuel consumption minimization or power consumption minimization. The use of $g = g_2(u) = u^TQu$, where $Q$ is the matrix given, suppresses the scope of control forces. If $g = c$, where $c$ is constant, the functional of quality represents the minimum time, which is

$$J(u)=\int_{t_0}^{t_f} c\,dt = c(t_f - t_0) \qquad [4.9]$$

Since $t_0$ is known, this functional quality will minimize $t_f$. This is the problem of time-optimal control.

### *4.2.4 Formulation of optimal control*

The aim of optimal control is to determine *u(t)*, which minimizes the *J(u)* functional. From the physical point of view state *x* shall be continuous; however, the control *u* can be interrupted. For better control performance, the control can need changing from its maximum value $U_i^+$ to its minimum value $U_i^-$. Such a moment is indicated as a switch over time. If the control is carried out only by the use of extreme values—the switch over between the minimum and maximum values—it is the so-called percussion control.

Optimal control means finding the permitted control *u(t)*, which means that the system (4.2) monitors the permitted trajectory *x(t)* and minimizes the functional quality (4.8). Such *u* and *x* are optimal control interventions and optimal state trajectories. Minimum $J(u) = J^*(u)$ means that:

$$J^*(u)=\int_{t_0}^{t_f} g(x^*,u^*,t)\,dt \le \int_{t_0}^{t_f} g(x,u,t)\,dt \qquad [4.10]$$

for all permitted states and all permitted control interventions. The values $J^*$, $x^*$, $u^*$ are optimal parameter values. They define the global minimum *J*. Inequality (4.10) can also be met only for some scopesof states *(||x|| < b)*, where || || means the standard of permitted trajectoriesand *b* is a positive value. In this case [4.10] would define the local minimum.

### *4.2.5 Types of optimal control*

Optimal control provides the history of permitted control interventions in the form $u(t) = f(x(t), t)$. From the point of the control, this is considered a control system with a closed loop, if it depends on the state. The law of optimal control can be linear, time-independent feedback, if $u(t) = Cx(t)$, where $C$ is a real matrix constant. The optimal control has an open loop, if $u(t) = f(t)$, as this does not depend on the state. The open regulation loop has several applications. An industrial robot manipulator with a specific task is an example with the open regulation loop.

## 4.3 Solution to optimal control

Optimal control similarly as all optimizing matters can be achieved via two methods: direct and indirect methods. The direct method is an approach, in which the sets $(x(k), u(k))$ and $(x(k+1), u(k+1))$, would be selected in two subsequent iterations so that $J(k+1) < J(k)$ . The functional of quality is directly minimized and simultaneously we try to meet all the restrictions via various exploration techniques. The direct methods usually utilize parametric optimization methods such as methods of punishment, gradient, associated gradient, etc.

Regarding the high number of parameters 7 as well as the time consumption, the direct methods (indicated also as parametric optimization solving random optimization matter) are quite inefficient.

An alternative approach is an indirect method. This method is more analytical than the direct method. The conditions to be met on the optimal path shall be derived as first. These conditions are represented by

Pontrjagin's Minimum Principle (PMP) and are essential for optimal solution. The further step is the determination of the controls and the trajectory meeting these conditions. In general, the indirect methods, in case of being successful, converge faster; however, it can come to convergence difficulties. They can be very complex in terms of mathematics as well. Due to this complexity, the indirect methods are now mostly used only to verify the solution found via other optimization methods.

To solve the optimization of complex systems controlling such manipulators, we need the numerical approach. Regarding the fact that PMP for time optimal control comprises initial and final conditions, the matter is two-pointed with marginal conditions. The shooting method is one of the basic ways to solve such tasks. Nevertheless, the method is very sensitive and converges by the optimal manipulator control.

## 4.4 Overview of existing methods of optimal motion control

Recently, there are lots of sources dealing with various aspects of optimal control. The fields considered can be classified into four larger groups, starting with general optimizing matters and going to more specific topics related directly to time optimal control of two- and more-armed manipulators.

The first group of contributions is focused on the optimal control stipulation from the point of view of vector optimization. The second group represents the optimal control application of flexible manipulators or structures. The third group comprises time optimizing matters of the control and their applications – which is also the topic of our contribution. The

fourth group describes the numerical methods for solution of various matters to optimal control.

### *4.4.1 Optimization with more criteria*

We usually need to investigate and optimize several aspects of the proposal process. This prepares optimizing tasks with more than one goal, which represents the vector optimization. The vector optimization includes the matter of optimal control and can be expressed as follows

$$J_i(u) = \int_{t_0}^{t_f} g_i(x,u,t)\,dt \rightarrow \min \qquad [4.11]$$

where $i = 1, \ldots m$ and $m$ is the number of the criterion to be optimized.

By more criteria or by the vector optimization we deal with the proposal vector of variables suitable for all restrictions and minimize the components of the purpose functions vector.

The existence of target conflicts is one of the characteristic properties of multi-criteria optimization, i.e. none of the solutions allows the current minimization of all targets. This is sometimes called a compromise of the functional. The matter is commonly reduced to scalar optimization by the stipulation of alternative matters or an alternative functional.

### *4.4.2 Time-optimal control*

Time optimal matters can be represented by the following functional of quality

$$J(u) = \int_{t_0}^{t_f} dt = t_f - t_0 \qquad [4.12]$$

when the final time $t_f$ is unknown. It is characteristic that in these matters the control is usually not continuous. We can consider two types: for known

trajectories it is necessary to find the control *u* (monitoring the path at the shortest possible time), or also the trajectory *u* shall be found (matter of time minimum).

### *4.4.3 Optimization via genetic algorithms*

Trajectory planning can be divided into two groups; one is planning along a defined path, and the other one is without the path given.

The space explored is reduced a lot for the matters related to the first category; therefore, they are the dynamic programming issues, graph methods, and phase plane algorithms. Phase plane algorithms are particularly efficient in time optimal planning. Nevertheless, it is complicated to apply these methods for high-dimensional exploring matters.

The matter in the other category is more complex than in the first one; both the path and trajectory planning have to be considered. This belongs to the matter of the two-point task in the optimal control theory and Pontrjagin's Minimum Principle provides us with basic analysis tools. The algorithm of shooting is a typical numerical tool for solving the problem. Other solution methods are based on the trajectory parameterization and non-linear programming. It is not simple to find the solution to the matter due to the non-linearity of the manipulator's dynamics. Therefore, the matter remains still unsolved.

Recently the genetic algorithms have performed as a suitable tool. The use of genetic algorithms has several advantages in comparison to classical methods. They have properties allowing them to avoid getting stuck in the local minimum and continue towards the global optimum via the combined information in many points of exploration, which makes GA robust in non-

linear matters. Further on, we will deal with the method of trajectory planning via genetic algorithms and its implementation with focus on the parameterization of genetic trajectory via acceleration.

### *4.4.4 Introduction to the subject matter solution via genetic algorithms*

In the genetic algorithm the population of strings is processed many times. Each element of the string represents a possible solution. Some strings represent unfeasible solutions, whereas some represent good solutions. Finally, after a long process, the population converges to the best possible solution, i.e. only copies of good solutions are left and the wrong ones are eliminated.

In our case the string should represent the nodes, which are the intersections in the motion trajectory of each joint. The best string or chromosome is the one that optimizes for example the motion time or overall electric power consumed by the manipulator.

Regarding the fact that we code directly the string of real numbers, the process is called a "Genetic algorithm with real coding".

Genetic algorithms begin with the initial population of individuals. The population is randomly initialized within the joint restrictions; the processes of selection, crossover and mutation help develop towards better and better fields when exploring the space.

**Matter representation**

We have a two-armed plane robotic manipulator which should move from the start position to the stable destination position. The aim is to find an optimal path by which the manipulator passes at the shortest possible time.

To simplify the matter we consider these prerequisites:

1. the robot is considered to be a two-armed plane manipulator;
2. individual kinematic restrictions shall be stated and tolerable trajectory points shall be from its possible working area;
3. the overall manipulator trajectory comprises transition points – nodes obtained from the genetic algorithm and are processed in regular time intervals;
4. for individual paths among the nodes, the approximation via the spline curve is used;
5. it is presumed that the manipulator end effector starts its motion from zero velocity and ends on zero velocity, while it does not stop on the transition node.

## 4.5 Formulation of genetic trajectory planning

Simple genetic algorithm usually uses binary coding for parameter representation. We consider the set of real parameters with $N_p$ number, which is given to x = {$x_1$, $x_2$,…,$x_i$,…, $x_{N_p}$ } (further we define it as x = $\{U_{i=1}^{N_p} x_i\}$), is coded into a binary string $\hat{x}$ (=$\{U_{i=1}^{N_p} \hat{x}_i\}$), and which is called a chromosome. Each real $x_i$ parameter having the maximum boundary value $x_i^U$ and minimum boundary value $x_i^L$ is coded into a binary string $\hat{x}_i$ using the binary length $\hat{L}_i$ .

For the manipulator shown in Fig. 4 it is necessary to optimize nine parameters in the form of the following chromosome:

$[q_1, q_2, q_3, q_g, \dot{q}_1, \dot{q}_2, \dot{q}_3, t_1, t_2]$

where

$q_i$ are angles of arm turns in transition points,

$\dot{q}_i$ are velocities of i$^{th}$ joint,

$q_g$ is a total angle of the final manipulator configuration, which equals to the addition of the angles,

$t_1$ is time from the start to the transition positions,

$t_2$ is time from the transition to destination positions.

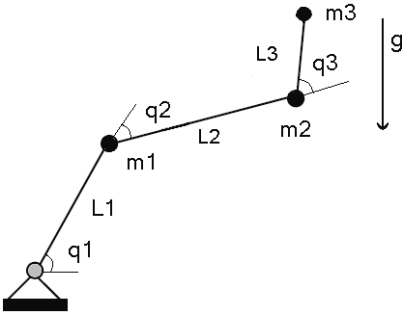


***Fig. 4*** *Manipulator with three links*

### *4.5.1 Fitness transformation via punishment function*

Similarly as many other engineering matters, planning the optimal trajectory of a robotic manipulator can be understand as a kind of optimization matter with a restriction. A genetic algorithm is not primarily

determined for this matter; however, this is solved by the introduction of the so-called finding function. The following purpose function, with the restrictions represented by the limitations of equality and inequality,

$$\text{minimize} \quad f(x) \qquad [4.13]$$

$$\text{with respect to} \quad g_k(x) \leq 0 (k = 1,2, \ldots, M_1)$$
$$h_l(x) = 0 (l = 1,2, \ldots, M_2).$$

We can convert to an assistant/auxiliary function without restrictions for the finding function in the following form:

$$min\ P(x) = f(x) + \sum_{k=1}^{M_1} w_k . \phi(g_k(x)) + \sum_{k=1}^{M_2} w_l . \psi(h_l(x)) \qquad [4.14]$$

where $M_1$ and $M_2$ are restriction numbers of inequalities and equalities. $\phi()$ and $\psi()$ are finding functions for inequalities and equalities restrictions, which are usually determined as $\phi(y) = |max(0,y)^m|$ and $\psi(y) = |y|^m$. $max(x, y)$ returns the maximum value between $x$ and $y$. $|.|$ means an absolute value of the function and $m$ is a positive number. In genetic algorithms the fitness is defined as a maximization of the purpose function and it shall be positive. A commonly used fitness transformation is the inverse value of the auxiliary function (4.14) or its subtraction from some high positive number $C_{max}$. Therefore, the fitness of the aforementioned issue can be expressed as follows:

$$fit = max\ (0,1/P(x)) \text{ or } max\ (0, C_{max} - P(x)) \qquad [4.15]$$

### *4.5.2 Subject matter definition*

For many industrial applications the current robotic manipulators are slow to be used economically. Their velocity and thus their productivity are limited by their drives' capability. The increase of the drives and their force is not the best solution, as the inertia of the drives themselves increases, as well as the price and power consumption. The minimization of time necessary for the execution of the given task regarding the drives' restrictions is a more successful approach.

There are more approaches to the issue, however I selected the method of genetic trajectory parameterization via acceleration (13), which regarding my research experience is the most elaborated method utilizing evolutionary principles and considering many aspects concerning the optimal motion of the manipulator, including its dynamics, in the environment with some obstacles or without. The method does not have the point approximation of the trajectory motion along the curve, which is important due to the manipulator motion fluency and due to avoidance of jump motions from point to point – this the subject of my improvement in Chapter 5.

First we get acquainted with the method and then I introduce the improvement implementation.

The essential idea of the method (13) is to select such an acceleration profile producing the highest velocity profile, so that for each path point the maximum velocity is not higher than the velocity by which the drives keep the manipulator on the track without breaking the restrictions.

We define the matter of planning the optimal trajectory of an industrial robot.

For given start and destination marginal trajectory conditions (OP):

$q(0) = q_0,\ q(t_F) = q_F$ {OPP}: position [4.16]

$\dot{q}(0) = 0,\ \dot{q}(t_F) = 0$ {OPR}: velocity [4.17]

and dynamics of the robotic manipulator:

$M(q)\ \ddot{q} + C(q, \dot{q}) + G(q) = T$ {R} [4.18]

to find optimal manipulator trajectories having the certain minimum criterion, in our case we consider time minimization:

$$\int_0^{t_F} dt = \boldsymbol{t_F} \quad \{\text{criterion}\} \qquad [4.19]$$

with meeting the following restriction conditions (ObP):

$Q^L \le q \le Q^U$ {ObPP}: position [4.20]

$V^L \le \dot{q} \le V^U$ {ObPR}: velocity [4.21]

$A^L \le \ddot{q} \le A^U$ {ObPZ}: acceleration [4.22]

$T^L \le T \le T^U$ {ObPM}: moment [4.23]

where

$n$ : manipulator degree of freedom,

$t_F$ : path time from the start to destination positions,

$q,\ \dot{q},\ \ddot{q} \in \mathrm{R}^n$ : generalized vectors of position, velocity and acceleration,

$T \in R^n$ : generalized vector of the moment (force) of the control,

$M(q) \in \mathrm{R}^{n \times n}$ : matrix of inertia,

$C(q, \dot{q}) \in R^n$ : Coriolis's and centrifugal vector of force,

$G(q) \in R^n$ : vector of gravitation force,

$^{U,L}$ : top and bottom border/boundary values.

### *4.5.3 Parameterization of genetic trajectory*

Variables of a trajectory can be divided into two groups: variables of a kinematic trajectory and a control moment. Variables of the kinematic trajectory are as follows: arm position, velocity and acceleration. Trajectory restriction consists of two parts: restrictions of equality of trajectory marginal conditions (OPP, OPR) and inequality of restricting trajectory conditions (ObPP, ObPR, ObPZ and ObPM).

In robotics it is important how to select the parameters from trajectory variables and how to select the optimization method for trajectory planning. We chose the arm acceleration as a parameter for trajectory genetic parameterization. Despite having chosen tangential acceleration as the variable for mathematical expression of time optimization, we also consider the drives of moments controlling the motion.

## 4.6 Procedure of genetic trajectory planner

### *4.6.1 Acceleration parameterization for genetic algorithm*

In this part I describe the procedure of acceleration parameterization for effective genetic algorithm implementation.

We introduce the following variables:

$n$: number of manipulator joints

$N$: number of trajectory parts

$\Delta t(=t_F/N)$: evenly divided path time

$q_j, v_j (= \dot{q}_j)$, $a_j (= \ddot{q}_j)$, $T_j$: position, velocity, acceleration and moment of the j[th] joint

$q_{j,i}$, $v_{j,i} (= \dot{q}_{j,i})$, $A_{j,i} (= \ddot{q}_{j,i})$ $(j = 1,2, ..., N)$: position, velocity, acceleration of the j[th] joint in the i[th] node point

$Q_{j,i}$, $V_{j,i}$ $(i = 0,1, ...., N)$: position, velocity of the j[th] joint in the i[th] node point where $i=0,N$ means start and destination node points

$Q = \{Q_{j,i} \mid j = 1,2,..., n, i=0,1,...,N\}$ : set of node points positions

$V = \{V_{j,i} \mid j = 1,2,..., n, i=0,1,...,N\}$ : set of velocities of node points

$A = \{A_{j,i} \mid j = 1,2,..., n, i=1,2,...,N\}$ : set of accelerations of node points

The method of trajectory discretion: First we divide the path time interval $[0,t_F]$ into $N$ amount of the same partial intervals. That is:

$$[0,t_F] = [t_0,t_1] \cup [t_1,t_2] \cup ,..., \cup [t_{N-1},t_N] \qquad [4.24]$$

where

$$\Delta t = t_i - t_{i-1} = \frac{t_F}{N} \ (i = 1,2,...,N) \qquad [4.25]$$

The accelerations remain constant in each partial path time interval, i.e.:

$$A_{j,i} = const. \ (i=1,2,...,N) \qquad [4.26]$$

For the explicit path time specification and its adoption as a system parameter, we express the path time $t \in [t_{i-1}, t_i]$ by a standardized $\tau$ parameter as follows:

$$\tau = \frac{t - t_{i-1}}{\Delta t} \quad (i = 1,2,..., N),\ \tau \in [0,1] \qquad [4.27]$$

Then the velocity of the joint in the i[th] partial interval of the path time can be expressed as follows:

$$v_{j,i} = V_{j,i-1} + \int_{t_{i-1}}^{t} A_{j,i}\,dt \qquad [4.28]$$

and the shift of the joint can be expressed as follows:

$$q_{j,i} = Q_{j,i-1} + \int_{t_{i-1}}^{t} v_{j,i}\,dt = Q_{j,i-1} + \frac{1}{2}\tau\Delta t(V_{j,i-1} + v_{j,I} \qquad [4.29]$$

The overall sum of the joints' accelerations and the related interval of the path time shall be the difference of the destination and start velocity which is zero in this case (OPR).

$$\int_{0}^{t_F} a_j\,dt = \sum_{i=1}^{N} A_{j,i}\,\Delta t = 0 \qquad [4.30]$$

Then the joint acceleration in the i[th] node point of the time path interval we obtain recursively as follows:

$$V_{j,i} = V_{j,i-1} + A_{j,i}\Delta t = \sum_{k=1}^{i} A_{j,k}\,\Delta t \qquad [4.31]$$

The joint position in the i[th] node point of the path time interval is given:

$$Q_{j,i} = Q_{j,i-1} + \frac{1}{2}\tau\Delta t(V_{j,i-1} + V_{j,i})\,\Delta t \qquad [4.32]$$

After substitution of (4.30) for (4.33) we get:

$$Q_{j,N} - Q_{j,0} = \sum_{i=1}^{N-1} V_{j,i}\Delta t = \sum_{i=1}^{N-1} (N-i) A_{j,i}\Delta t^2 \qquad [4.33]$$

After the substitution, the relation among the joint position and two points and joint accelerations is as follows:

$$Q_{j,N} - Q_{j,0} = N(\sum_{i=1}^{N} A_{j,i} - A_{j,N})\Delta t^2 - \sum_{i=1}^{N-1} iA_{j,i}\Delta t^2 = - \sum_{i=1}^{N} iA_{j,i}\Delta t^2 \qquad [4.34]$$

If we consider the joints' accelerations as genetic coding parameters, then two dependent parameters of joint accelerations $A_{j,k}$ and $A_{j,l}$ shall be sufficient for the equalities (4.30) and (4.34) to meet two marginal trajectory conditions (OPP, OPR). That is:

$$\begin{bmatrix} A_{j.k} \\ A_{j,l} \end{bmatrix} = \frac{1}{l-k} \begin{bmatrix} l & -1 \\ -k & 1 \end{bmatrix} \cdot \begin{bmatrix} -\sum_{i=1, i\neq k,l}^{N} A_{j,i} \\ \frac{Q_{j,0} - Q_{j,N}}{\Delta t^2} - \sum_{i=1, i\neq k,l}^{N} iA_{j,i} \end{bmatrix}$$

for $k, l = 1,2,..., N$ and $k \neq l$.

### *4.6.2 Trajectory parameter coding*

If we take the acceleration as a parameter of genetic trajectory coding, in general, we can select two dependent parameters $A_{j,N-1}$, $A_{j,N}$ to meet the marginal trajectory conditions (OPP, OPR). The set of coding parameters of each of the individual strings is given as follows:

$$\hat{x} = \left\{ \bigcup_{j=1}^{n} \bigcup_{i=1}^{N-2} A_{j,i} \;\; a \;\; t_F \right\} \qquad [4.35]$$

where $\left\{ \bigcup_{j=1}^{n} \bigcup_{i=1}^{N-2} A_{j,i} \right\}$ are coding parameters of acceleration and $t_F$ is the coding parameter of the path time. Two dependent parameters of acceleration selected with respect to marginal trajectory conditions (OPP, OPR) are given as follows:

$$\begin{bmatrix} A_{j.N-1} \\ A_{j,N} \end{bmatrix} = \begin{bmatrix} N & -1 \\ 1-N & 1 \end{bmatrix} \cdot \begin{bmatrix} -\sum_{i=1}^{N-2} A_{j,i} \\ \dfrac{Q_{j,0} - Q_{j,N}}{\Delta t^2} - \sum_{i=1}^{N-2} i A_{j,i} \end{bmatrix} \qquad [4.36]$$

The size of the coding parameter is for each individual string $\hat{x}$ as follows:

$$N_p = n.(N-2) + 1 \qquad [4.37]$$

### *4.6.3 Working with limit conditions*

It is not easy to work with the limit conditions of the robotic manipulator trajectory expressed by the equations of inequalities; therefore we transformed them to static restrictions. For illustration, we consider the following dynamic system:

we have: $\dot{x}(t) = f(x(t), u(t))$

for which it applies: $x^L \le x(t) \le x^U, \;\; \forall t \in [0, t_F]$ [4.38]

where $x(t)$ is a state variable with bottom boundary $x^L$, top boundary $x^U$ and $u$ is the controlled input. Continuous dynamic restrictions in the form of

inequalities (4.38) can be transformed to static restrictions in the form of inequalities:

$$c(x) = \int_0^{t_F} w_x \left(\min\left(x^U - x, 0\right)\right)^2 + \left(\min\left(x - x^L, 0\right)\right)^2 dt \qquad [4.39]$$

If $c(x)$ is equal to zero, the inequality restrictions in (4.38) is met. Similarly, we will work with the restrictions of trajectory limits. For simplicity, we consider $x_j$ as trajectory variable of such a j[th] joint, that $x_j = q_j$ *(position),* $v_j$ *(velocity),* $a_j$ *(acceleration),* $T_j$ *(turning moment)*. And we indicate the trajectory variable from the record (4.20) – (4.23) as $x_j(t) \in \left[x_j^L, x_j^U\right], \forall t \in \left[0, t_F\right]$, where $x_j^L$ and $x_j^U$ are the bottom and top boundaries of each trajectory variable of the j[th] joint. Then we can rewrite (4.20) – (4.23) as follows:

$$G(x_j) = W_x^t \, g(x_j) \qquad [4.40]$$

where

$$g(x_j) = \begin{bmatrix} \int_0^{t_F} \left|\min(1 - x_j / x_j^L, 0)^m\right| dt \\ \int_0^{t_F} \left|\min(1 - x_j / x_j^U, 0)^m\right| dt \end{bmatrix} \in R^{2\times 1}$$

is the vector breaking the restrictions, $W_x^t = \left[w_x^L \; w_x^U\right] \in R^{1\times 2}$ is the weight vector related to the bottom and top boundaries $x$ and $m$ is a positive number of the exponent. If $G(x_j)$ is approaching zero, i.e. that, $x_j$ trajectory variable meets its restriction condition in [4.20] – [4.23].

Fighting off the trajectory restriction condition in (4.40) to the fitness function, the fitness trajectories for trajectory genetic planning are indicated as:

$$fit = max\left(0, \frac{1}{t_F + \sum_x \sum_{j=1}^{n} G(x_j)}\right) \quad [4.41]$$

where

$t_F$ : time of path (criterion of minimization),

*max(x,y)*: maximum value between *x* and *y*,

$x_j$ : type of variable trajectory of such a j[th] joint, where $x =$ *q(position), v(velocity), a(acceleration), T (turning moment)*,

*G(x)* : modified limit conditions related to *x* type (ObPP, ObPR, ObPZ, ObPM),

$\sum_x$ : record expressing every limit condition in the fitness trajectory.

# 5. SOLUTION PROPOSAL AND IMPLEMENTATION

## 5.1 Algorithm of the whole procedure

The procedure for genetic planning of the trajectory for the robotic manipulator is expressed as follows:

$\hat{x} = \{ \hat{x}_i \mid i = 1,2, \ldots , N_p \}$ : trajectory chromosome in [4.35]

$k = 1,2, \ldots, N_{pop}$ : index of k$^{th}$ population individual

$\hat{X} = \{ \hat{x}^k \mid k = 1,2, \ldots , N_{pop} \}$ : population chromosome

$Fit = \{ fit^k \mid k = 1,2, \ldots , N_{pop} \}$ : fitness population vector

$\Omega_s : \hat{X} \times R^{N_{pop}} \mapsto \hat{X}$ ; selection operator

$\Omega_c : \hat{X} \mapsto \hat{X}$ ; crossover operator

$\Omega_m : \hat{X} \mapsto \hat{X}$ ; mutation operator

**Initial conditions:**

robot: arms parameters

$Q^L, \ldots , T^U$ : bottom and top boundaries of limit conditions in (4.20) – (4.23)

GA: $p_c, p_m \in [0,1]$ : probability of crossover, mutation

$[ \hat{x}^L , \hat{x}^U ]$ : $\hat{x}$ chromosome boundaries coding

$\hat{L} = \{ \hat{L}_i \mid i = 1,2,..., N_p \}$ : lengths of coding $\hat{x}$

$N_{pop}$ : population size, $N_{gen}$ : max. number of generations

other: $d\tau$ : period of sampling, $N$ : partial path time

Input: $q_0$, $q_F$ : start and destination robot positions (OPP)

Stop:

$\Lambda : \hat{X} \mapsto$ {Yes, No}; stop when the generation number achieves $N_{gen}$

outcome:

trajectory of an elite string meeting the stop criteria

**Pseudocode Algorithm**

1. Initialization
   1.1. Execute start setting of robot's parameters and genetic algorithm
   1.2. Input conditions (OPP)
   1.3. *gen* $\leftarrow$ *1,* $\hat{X}$ *(gen)* $\leftarrow$ Initialization of chromosomes ( $\hat{x}^L$ , $\hat{x}^U$ , $\hat{L}$ , $N_{pop}$)

*while* ($\Lambda : \hat{X}$ *(gen)* $\neq$ *Yes*) *do*

2. Evaluation of *Fit(gen)* s $\hat{X}$ (gen)
   *for k=1 to* $N_{pop}$ *do*
   *2.1.* $\hat{x} \leftarrow \hat{x}^k$
   2.2. calculation of $\Delta t$ according to [4.25]
   2.3. calculation of the set of accelerations $A$ according to [4.36]
   2.4. calculation of the set of speeds of $V$ nodes, of the set of positions $Q$ according to [4.31] – [4.32]
   2.5. calculation of $\ddot{q}$ acceleration from $A$, $\dot{q}$ velocity and $q$ position according to [4.28] – [4.29] and $T$ moment according to [4.18]
   2.6. calculations of fitness *fit* according to [4.40] – [4.41]
   *2.7.* $fit^k \leftarrow fit$

*end*

3. GA operations

3.1. $\hat{X}_s(gen) \leftarrow \Omega_s(\hat{X}(gen), Fit(gen))$

3.2. $\hat{X}_c(gen+1) \leftarrow \Omega_c(\hat{X}_s(gen))$

3.3. $\hat{X}_m(gen+1) \leftarrow \Omega_m(\hat{X}_c(gen))$

4. $gen \leftarrow gen + 1,\ \hat{X}(gen) \leftarrow \hat{X}_m(gen)$

*end*

*return the result*

Kinematic chains of industrial robots are usually open and consist of two parts. The first part is a positioning device comprising individual components and shifting and rotation kinematic couples. The other part of the industrial robot kinematic chain is the device for orientation consisting mainly of rotation kinematic couples with one, two, or three motion degrees of freedom. A mechanical system of an open industrial robot kinematic chain is accomplished by an effector, i.e. an executive link of the robot.

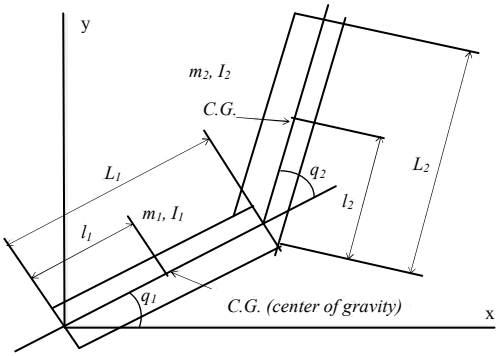


***Fig. 5*** *Kinematic scheme of an industrial robot*

The position of the *P* working point of the robotic effector regarding the coordinate system of the mechanism and for the orientation according to Fig. 5 can be expressed as follows:

$x_P = L_1.sin\ q_1 + L_2.sin\ (q_1 + q_2)$

$y_P = L_1\ cos\ q_1 + L_2\ cos\ (q_1 + q_2)$

The task is to investigate the necessary motion in kinematic couples to ensure the replacement of the *P* working point of the robot effector from $P_0$ position to $P_1$ position. At the beginning and the end of the motion the speed and acceleration of the *P* point shall be zero.

Dynamic equations of the manipulator in Fig. 4 can be stated as follows:

$$T_1 = M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 - 2h\dot{q}_1\dot{q}_2 - h\dot{q}_2^2$$

$$T_2 = M_{22}\ddot{q}_2 + M_{12}\ddot{q}_1 + h\dot{q}_1^2$$

where

$$M_{11} = I_1 + I_2 + m_1 l_1^2 + m_2\left(L_1^2 + l_2^2 + 2L_1 l_2\cos(q_2)\right)$$

$$M_{12} = I_2 + m_2 l_2^2 + m_2 L_1 l_2\cos(q_2)$$

$$M_{22} = I_2 + m_2 l_2^2$$

$$h = m_2 L_1 l_2\sin(q_2)$$

while

$M_{11}, M_{12}, M_{22}$ are drive moments generated by servo gears $L_1, L_2$ are lengths of robot links

$I_1, I_2$ are mass moments of inertia to link centres

$l_1, l_2$ are distances of individual links centres

$m_1, m_2$ are masses of individual links.

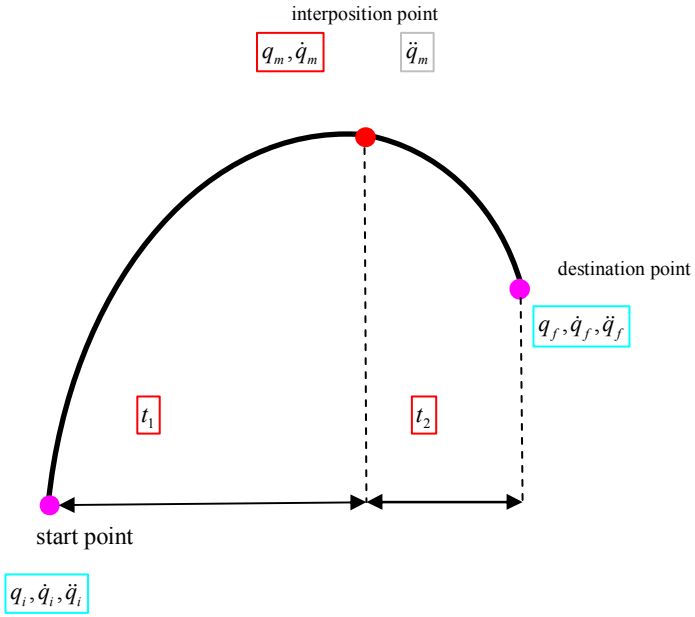


***Fig. 6*** *Destination effector point trajectory*

We look for time courses of turn angles $q_i(t)$ and $q_j(t)$ in the form of a fifth degree polynomial:

$$q_i(t) = a_1t^5 + a_2t^4 + a_3t^3 + a_4t^2 + a_5t + a_6$$

$$q_j(t) = b_1t^5 + b_2t^4 + b_3t^3 + b_4t^2 + b_5t + b_6$$

Constants in these polynomials are determined from the start and destination conditions for the motion of *P* working point.

The point *P* trajectory shown in Fig. 6 corresponds with the replacement from the start to the destination positions.

Turn angles of individual mechanism components - for the orientation expressed as functions of time – represent the kinematic control functions of the robot's mechanical subsystem.

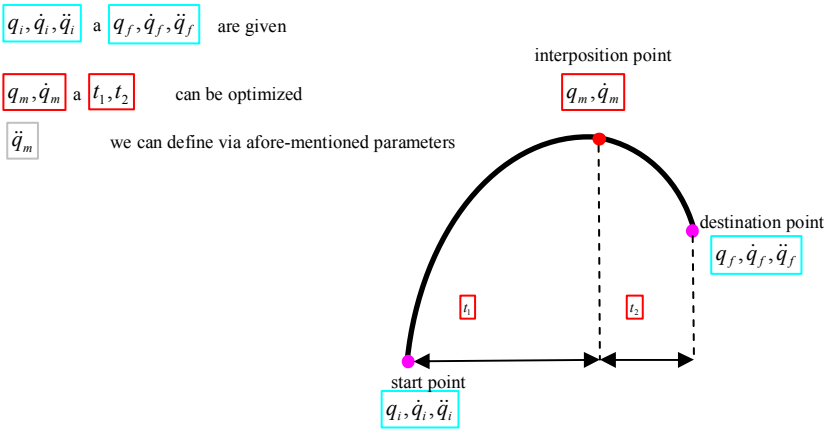


***Fig.** 7 What is needed to optimize*

## 5.2 Solution implementation in support software

### *5.2.1 Robot model*

To understand what properties the unoptimized trajectory has, it is suitable to prepare a robot model respecting its kinematics and dynamics. We looked for the suitable environment, in which the implementation could be executed for quite a long time. We selected MATLAB™, which allows a lot of engineering and scientific calculations, and it is possible to supplement it by various toolboxes, programming of one's own functions, and therefore its possibilities in the field of scientific calculations are almost unlimited.

We built the robot model in the MATLAB™ program with the use of the RobotiCad Toolbox which allowed us to create the motion scheme of a random kinematic chain in a user-friendly way. It also cooperates with the SIMULINK® simulation tool, where the dynamic chain properties can be defined as necessary for the application of our method of optimal trajectory generation.

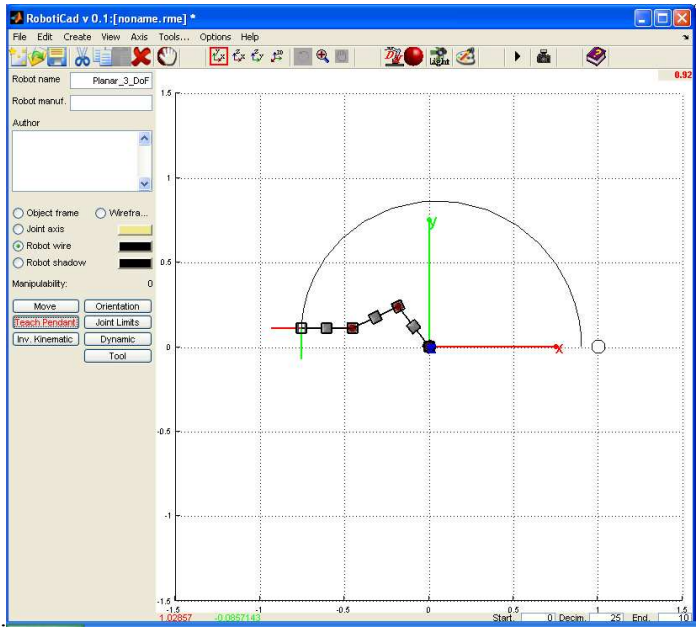


***Fig. 8*** *Plane mechanism trajectory*

Fig. 8 shows the model of a plane mechanism with three degrees of freedom. The program allows illustrating the destination effector trajectory. For illustration we chose a simple trajectory from the point with zero *y* coordinate, positive *x* coordinate and zero arms turn angles to the point with

a negative $x$ coordinate, non-zero $y$ coordinate and positive arms turn angles.

We would like to optimize a similar trajectory in the MATLAB™ Optimization Toolbox. The robot in real applications carries out a lot of motion actions, so its trajectory comprises a lot of smaller trajectories similar to the one we have decided to analyze. The principle applies for any random trajectories.

First we illustrated the time courses of the angles of the arms´ turns, speeds and accelerations in order to show the trajectory given and describe what we need to improve.

The horizontal axis comprises the time; the vertical axis comprises the angle (Fig. 9), speed and acceleration.

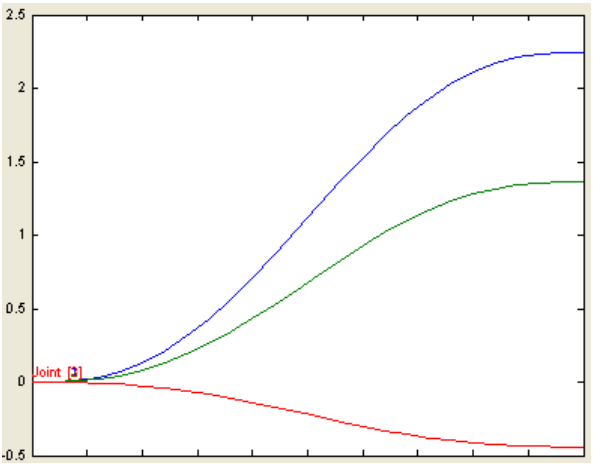


***Fig. 9*** *Time courses of arms angles*

The illustrated courses of individual turn angles in Fig. 9 show that the curves do not indicate any “interventions” during the motion, which could lead to possible more efficient motion during the manipulation action. The same applies for the courses of the arms´ velocities and accelerations in Fig. 10 and Fig. 11.

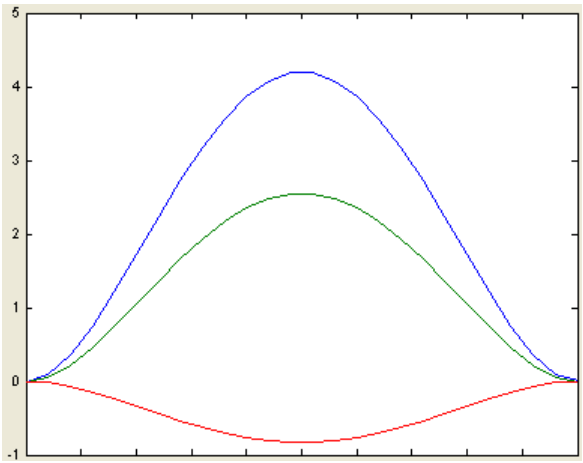


***Fig. 10*** *Time courses of arms' velocities*

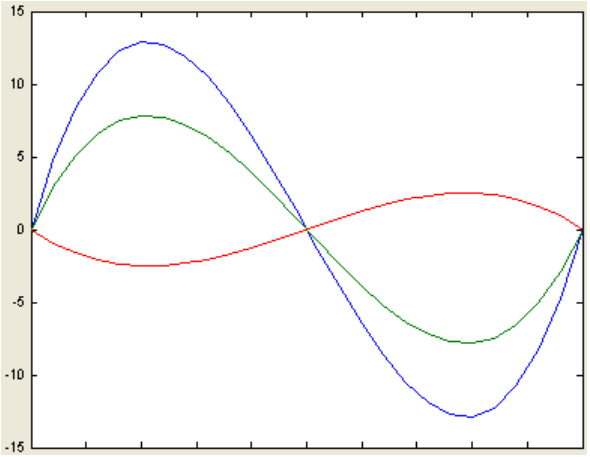


***Fig. 11*** *Time courses of arms' accelerations*

### *5.2.2 Trajectory optimization*

In this chapter I describe the parameters of the plane mechanism tested. The arms' lengths were $l_1$ = 1 m, $l_2$ = 1 m and $l_3$ = 0.5 m. Mass $m_1$ = 1 kg, $m_2$ = 1 kg and $m_3$ = 0.5 kg. The maximum permitted drive moments 1, 2 and 3 are 45 Nm, 20 Nm and 5 Nm. The speeds and accelerations in the start and destination positions are null.

For the genetic algorithm the following parameters were valid: crossover probability $P_c$ = 0.8 to a chromosome (function $P_c$ determines how often the chromosome is crossed), mutation probability $P_m$ = 0.05 (function $P_m$ determines how often than chromosome part is mutated) and the population of 40 individuals for the angles in the interpositions, arms´ speeds and times, string size of the chromosome 9. The number of crossed chromosomes in each of the generations is defined as a multiple of $P_c$ and the population size. The number of mutated genes in each generation is

defined as a multiple of $P_m$, population size and chromosome length. We used a tournament selection, elitism and the maximum number of generations – 80.

We have carried out the following experimental results verification in the MATLAB™ environment with the use of a toolbox for optimization via genetic algorithms.

Fig. 12 shows the optimized trajectory, which even at first sight has a different trajectory course than the original one illustrated in Fig. 8. We can see the illustration of the gradual motion of individual arms; the turn angle of the first arm does not change first at all, and only in the third of the motion sequence a deviation can be recorded.

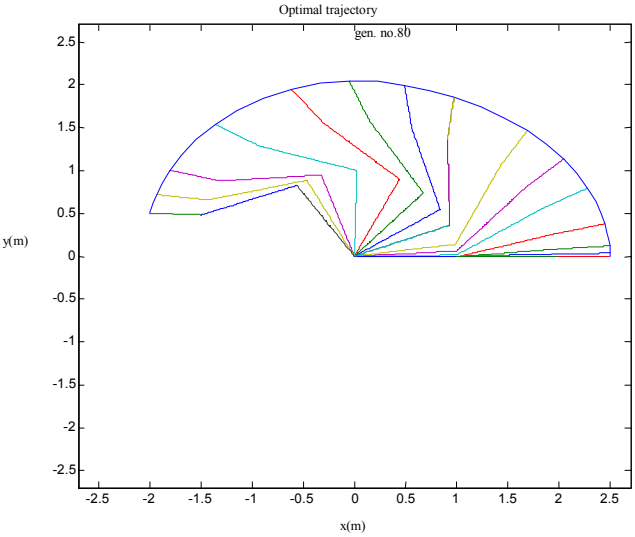


***Fig. 12*** *Optimized trajectory*

The largest turn angle and its largest moment of servo gear have been developed in the second joint immediately after the motion sequence beginning.

The illustrated course is calculated after the run of maximum number of generations - 80. During the calculation of individual generations the course has gradually approximated optimal values.

Fig. 13 shows the course of fitness in individual generations. We can see the falling tendency with the number of generations, which indicates the successful calculation leading to finding the time optimal parameters of the mechanism motion trajectory.

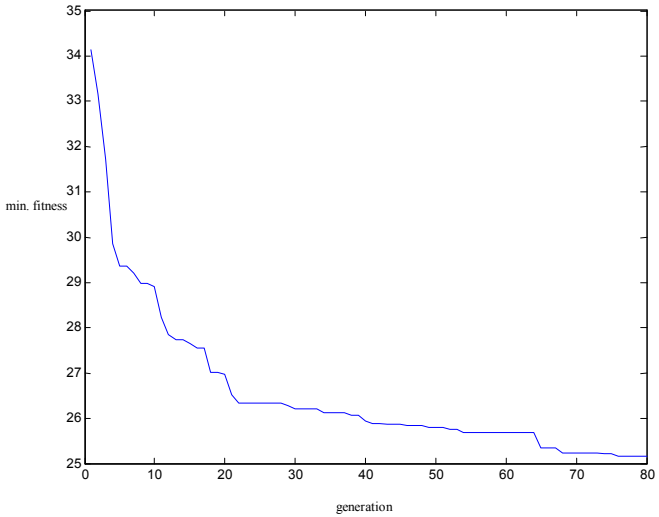


***Fig. 13*** *Course of fitness*

To compare, in Fig. 14 we can see the time courses of arm turns angles. In contrast to the course shown in Fig. 9 we can see a significant change in the angle turn of the other arm, which slows down in the other half of the motion sequence and the turn angle is almost not changed. The course of the angle turn of the first arm is different, it starts with an easy angle change and approximately in the half of the motion sequence it increases. The third arm is turned gradually almost evenly and continuously during the motion sequence.

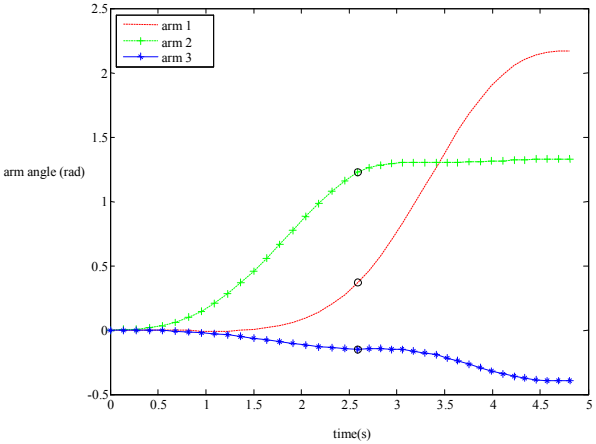


***Fig. 14*** *Time courses of arms angles*

In relation to the time courses of angles the time courses of other quantities change also, the velocities and accelerations of arms in particular. Fig. 15 illustrates the time courses of arms velocities.

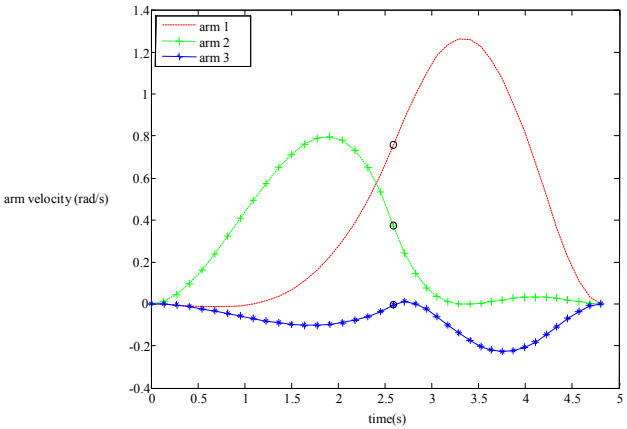


***Fig. 15*** *Time courses of arms velocities*

It is necessary to notice that the velocity of the other arm began to grow sharply in the first third of the manipulation sequence. In the second third it had a falling tendency and in the last one it was almost none. The first arm achieved the highest velocity in the last third of its motion.

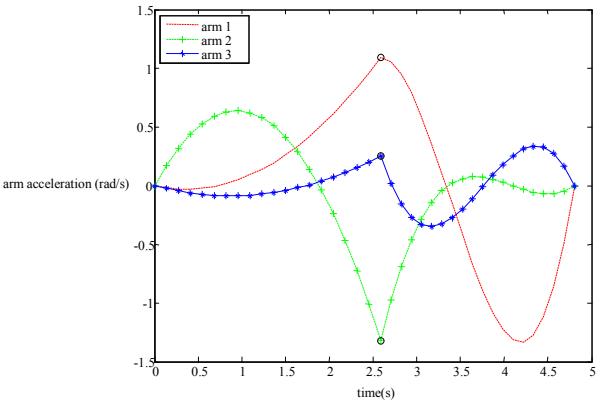


***Fig. 16*** *Time courses of arms' accelerations*

The courses of acceleration in Fig. 16 show that the gear properties for achieving the maximum allowed arms moments are utilized at their maximum. If we compare these courses to those in Fig. 11 from the original mechanism, the main difference, besides completely different curve shapes, is seen approximately in the middle of the motion, where a sharp course change occurs.

The main task is to define the moment when the breaking should occur. It is clearly seen in the course of the first arm, where the breaking moment occurs approximately in the middle of the motion.

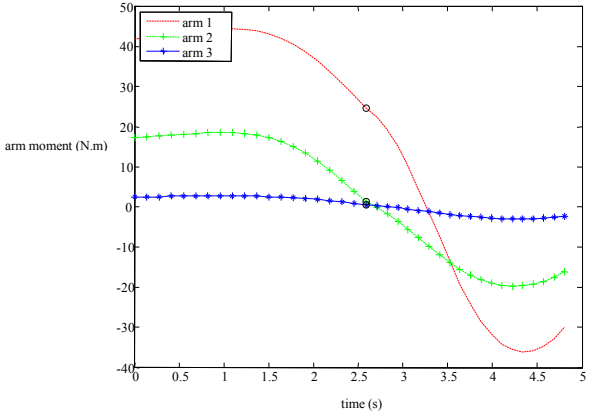


***Fig. 17*** *Time courses of arms' moments*

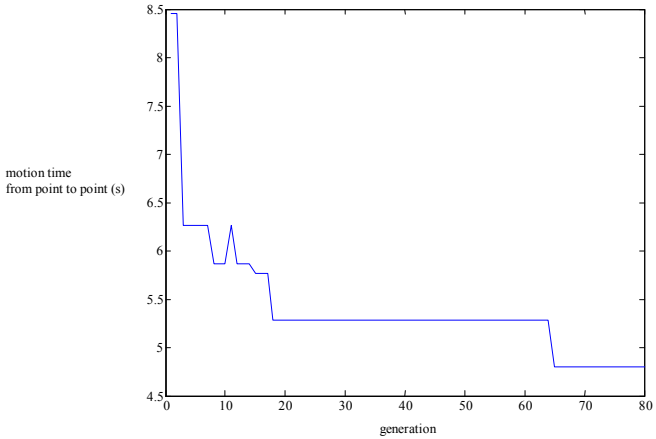


***Fig. 18*** *Dependence of motion duration from the point to the point on the number of generations*

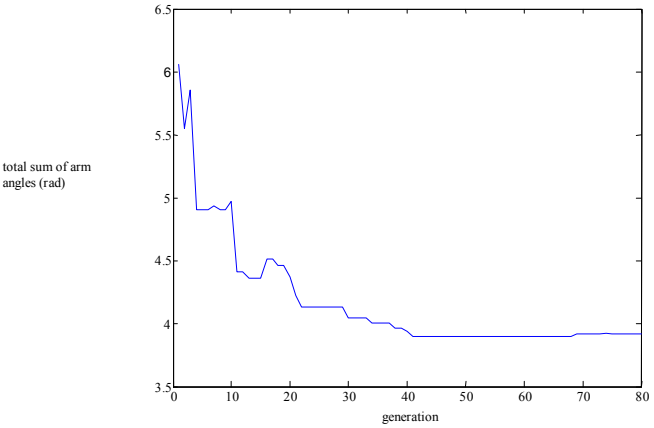


***Fig. 19*** *Dependence of the total sum of arms' turn angles on the number of generations*

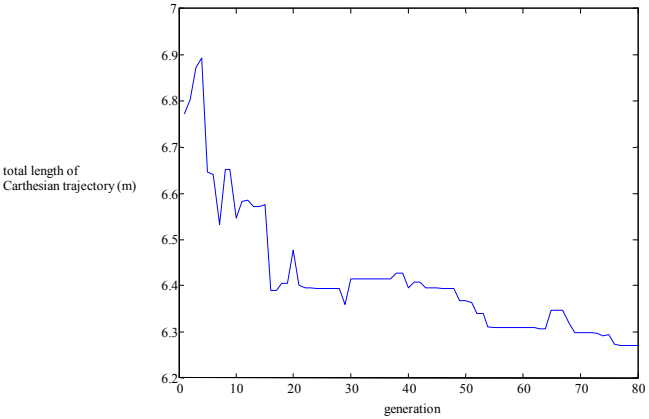


***Fig. 20*** *Dependence of the total length of Cartesian trajectory on the number of new generations*

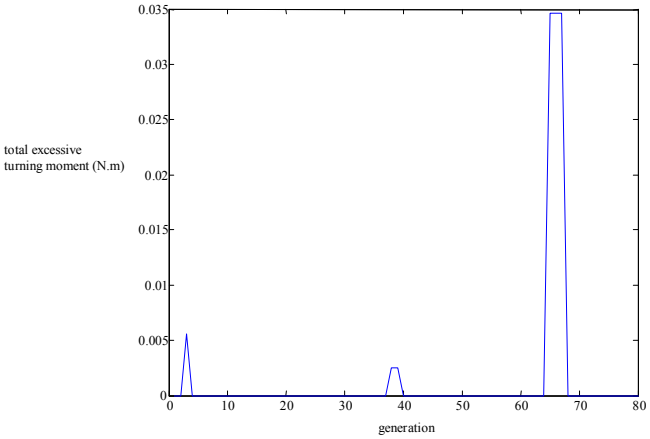


***Fig. 21*** *Dependence of the total excessive turning moment on the number of generations*

### *5.2.3 Simulation results*

The verifying study has been carried out for a 3-link plane mechanism moving in a free working space. We have learned the decrease of motion time and at the same time the decrease of Cartesian trajectory length.

Genetic algorithms have proved to be a suitable optimization method; the results are illustrated in the graphs in the previous chapter.

The moments of drives in the motion have not exceeded the allowed values and at the same time they have been fully utilized.

## 5.3 Experimental verification of results

Since the solution is not limited only to a specific manipulator type, it is possible to test it on a random robot type with regard to the setting of similar conditions as by the simulation mentioned. To test the optimized trajectory we have selected a real tool for an offline robot programming. As we have had the possibility to work with Denso robot (in Fig. 22), testing has been carried out in the environment delivered with it. It is WINCAPS III software, whose trial version has been at our disposal. WINCAPS III Program is a program package for an effective development and verification of robot control programs. It allows checking the robot operations, variables, PC inputs and outputs connected to the robot control circuits. It also allows program administration as projects, storing of frequently used programs in program files registers and also other functions for program functions administration. Denso Robot can be connected to a PC via Ethernet or via a serial port.

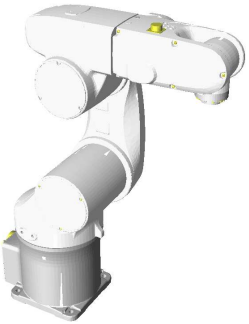

***Fig. 22*** *Denso VS-6556G Robot*

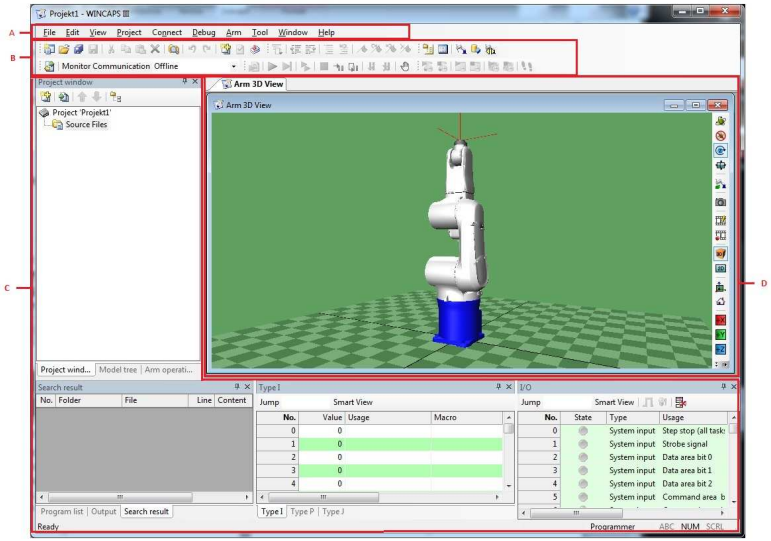


***Fig. 23*** *WINCAPS III Program environment*

Fig. 23 shows the WINCAPS III Program environment. In Part A there is the program menu, B comprises of various control panels, C is the so-called docking window with various information on the current project of a robotic workplace, etc., and D is the so-called view of the program where we can see the source code of the program, the 3D window with the robotic workplace, etc. A specific robot model from the robot models catalogue can be included to the program and then programmed. In our case it is the Denso VS-6556G robot.

In the Model window a 3D object representing the workplace elements or obstacles can be inserted into the scene.

In the Arm operation window we can define the robot's motion either in the mode of setting the point coordinates or in the mode of setting the individual arms' turn angles.

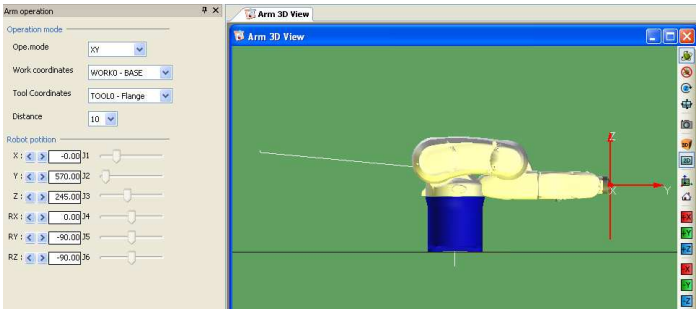


***Fig. 24*** *Start point in Arm operation*

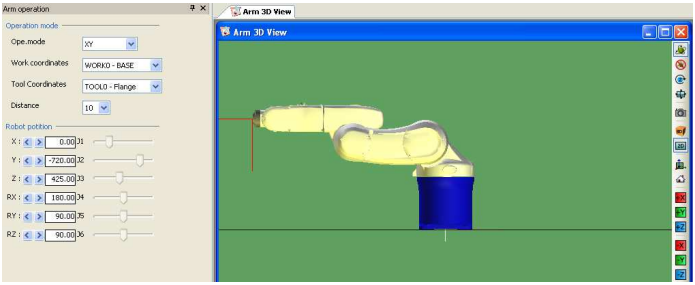


***Fig. 25*** *Destination point in Arm operation*

First I defined a common trajectory from one point to the other point in the Arm 3D View window.

In the bottom part of WINCAPS environment there is a window, which can take over the robot's defined position from the Arm operation window by clicking the key Get Position. Fig. 26 shows the defined positions of the robot in the destination point coordinates system mode. Line 0 corresponds with the start position and line 1 corresponds with the destination position of a robot. Fig. 27 shows the defined robot's positions in the mode of the arms' turn angles. Line 0 corresponds with the start position and line 2 with the destination position of a robot.

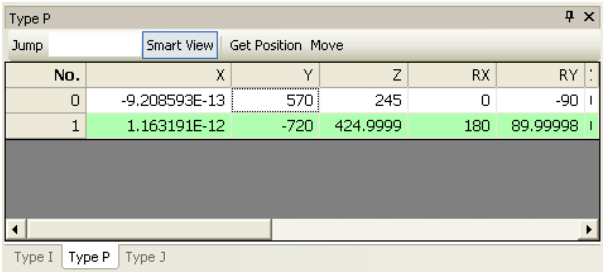


***Fig. 26*** *Defined points of a robot in the position coordinates mode*

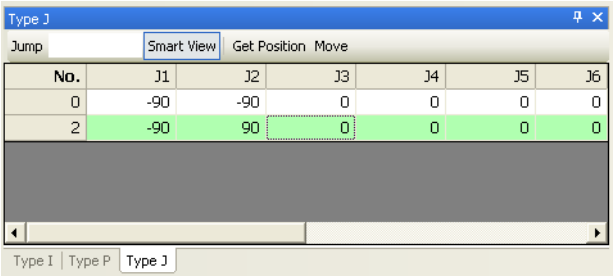


***Fig. 27*** *Defined points of a robot in the arms' turn angles mode*

To execute our experiment for verifying the simulation results in the MATLAB™ environment we define for the Denso mode robot the trajectory via motion points. This can be done in the ArmPlayerPlus window (Fig. 28), where we build the robot program by taking over the defined points by the order MOVE in the right side. Beside this, under the key MOVE, there are various ways of trajectory points definition, in the program it is displayed in the record of the order MOVE P, @parameter, where "parameter" means in the case:

- @P, that the robot in the given trajectory point does not accelerate and slow down, which on the other hand can result in the fact, that it does not cross the point all the time and can bypass it,
- @0 slows down and accelerates, but only partially; the point is partially bypassed,
- @E means complete stop in the point given; this is used if we need to reach an exact specific point.

Example of a simple program:

```
'!TITLE "<Titile>"
PROGRAM motion
        MOVE P,@P J4, S=100
        MOVE P,@E J5, S=100
END
```

Order MOVE P, @P J4, S=100 means that the robot executes a motion in the mode from the point to the point, while the point is determined by the arms turn angles from the Table type J in the line 4 (J4). Since @P is stated there, the robot does not accelerate nor slowdown in the point. In the end of the line the speed is defined in percentage (S=100), i.e. the robot moves to this point with a maximum possible speed.

Order MOVE P, @P J5, S=100 means that the robot carries out a motion in the mode from a point to a point, where the point is determined by the arms turn angle from the Table J in line 5 (J5). Since @E is stated, the robot stops in the point given.

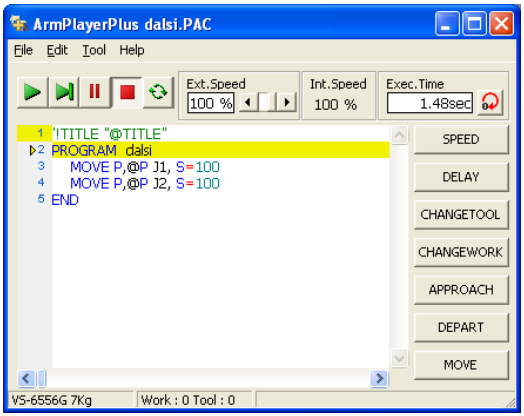


***Fig. 28*** *Definition of motion points in ArmPlayerPlus*

When we run the motion sequence in ArmPlayerPlus, the program will move the robot from the first point to the other point.

In the ArmPlayerPlus window we can see the time in seconds 6.83 s (Fig.29). It is the time in which the robot executes the motion from the starting point defined in line 21 to the destination point in line 30 (in the program specified as J30), while the motion is not optimized at all.

The trajectory is in the shape of an arc as illustrated in Fig. 8. The trajectory points are defined by well proportioned division of turn angles in the scope from -72° to 90°.

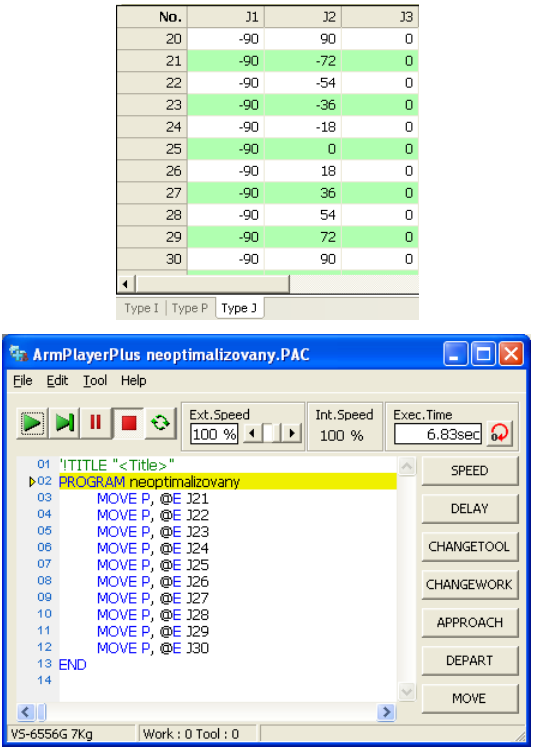


***Fig. 29*** *Coordinates of points and duration of a non-optimal trajectory*

Since we do not have available software to transfer the trajectory directly from the MATLAB™ environment to WINCAPS with defined trajectory points and speeds in individual points, it is necessary to carry out the action manually and set the optimized trajectory in the form of a specific number of trajectory points.

The table of points of the optimized trajectory is shown in Fig. 30. The points are given according to the generated optimal trajectory based on the data in Fig. 24 showing the course of the individual arms turn angles in specific time periods.

| No. | J1 | J2 | J3 |
|---|---|---|---|
| 10 | -90 | -90 | 0 |
| 11 | -90 | -85 | 15 |
| 12 | -90 | -80 | 25 |
| 13 | -90 | -70 | 45 |
| 14 | -90 | -65 | 75 |
| 15 | -90 | -30 | 85 |
| 16 | -90 | -2 | 70 |
| 17 | -90 | 30 | 45 |
| 18 | -90 | 54 | 15 |
| 19 | -90 | 72 | 0 |
| 20 | -90 | 90 | 0 |

Type I | Type P | Type J

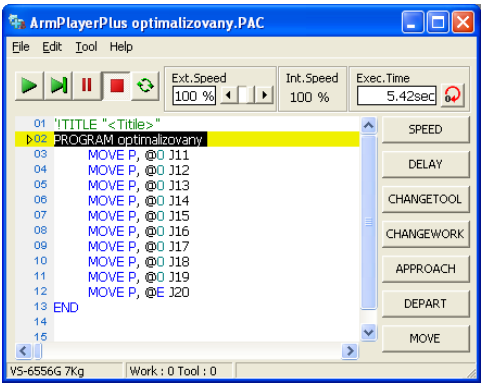


***Fig. 30*** *Coordinates of points and duration of the optimized trajectory*

After the assignment of the optimal trajectory due to the table of points defined in lines from 11 to 20 we have learned, that the generated optimized trajectory has shortened the time of motion to 5.42 s (Fig. 31). It is the time necessary for the robot to move from the start point defined in line 11 to the destination point defined in line 20 (in the program indicated as J20).

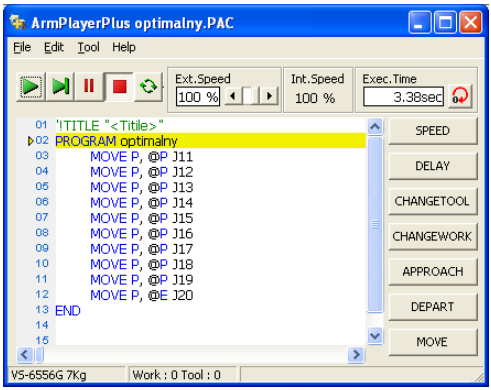


***Fig. 31*** *Duration of the optimal trajectory*

The method improvement proposed by us (13) in the form of the points' approximation of the motion trajectory along the curve has made the time shortening from 5.42 s to 3.38 s possible. The shortening of time is significant, since in the case of optimal trajectory method (13) being in individual points the manipulator carries out partial motions, slows down and subsequently accelerates to reach the point specified.

In the case of optimal trajectory it is a continuous motion along the curve whose parameters were generated by the applied genetic algorithm. Individual points of the existing trajectory are approximated by the curve,

no slowing down or acceleration occur, and therefore, the motion is continuous and the trajectory length is shorter.

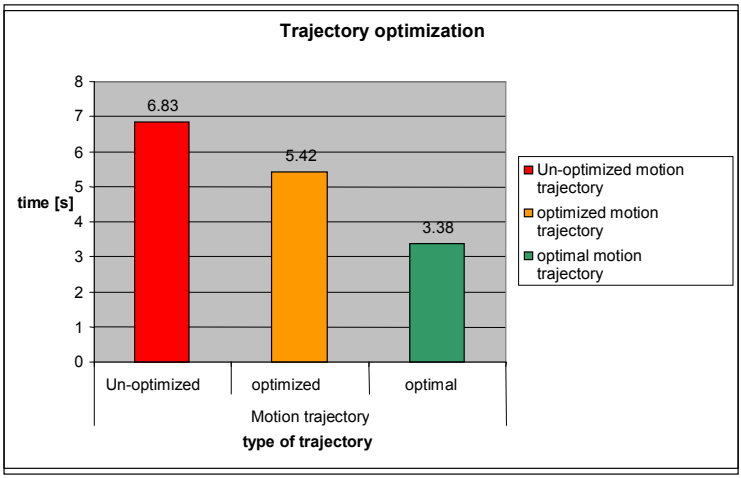


***Fig. 32*** *Trajectory optimization*

# 6. CONCLUSION

Based on the analysis of various existing methods and regarding the drives moments constraints, I implemented and improved the algorithm to an off-line generating of a time optimal trajectory generated via the genetic algorithm.

For a specific manipulator this algorithm requires: that the angles in terms of the position on the path can be calculated, the dynamic equations are known as well as the maximum and minimum possible generated drives moments as functions of arms angles, and angular speeds are known.

The algorithm implementation of the manipulator trajectory genetic planning which optimally controls the robot motion in terms of time is the main contribution. The method can be utilized to generate optimal parameters for an industrial robot motion trajectory with various numbers of freedom degrees.

For simplification and better illustration the experiments were carried out on the model of a surface manipulator with three freedom degrees. The mentioned procedure is suitable for further 3D processing, for other manipulator types, and testing of generated trajectory in some of the environments for a robot control in the mode of setting the points' motion coordinates.

It could be suitable to develop support software able to rewrite the generated trajectory from the MATLAB™ environment in the form of appropriate orders directly for a specific robot in a specific control environment.

Similarly, I considered only time as a criterion for minimization; however, this could be enhanced to further criteria such as power. Also other constraints and tasks, e.g. avoiding obstacles and cooperation of more robots could be subject to further research.

## References


1. LAVALLE, S. M. *Planning Algorithms.* University of Illinois, 2004.
2. SEKAJ, I. *Evolučné výpočty a ich využitie v praxi.* (*Evolutionary calculations and their utilization in practice.*) Bratislava: IRIS, 2005. 157 p. ISBN 80-89018-87-4
3. JURIŠICA, L., HUBINSKÝ, P, KARDOŠ, J. *Robotika.* (*Robotics.*) Bratislava: STU, 2005. 134 p.
4. CRAIG, J.J. *Introduction to Robotics.* Massachusetts: Addison·Wesley, 1986.
5. FU, K.S., GONZALEZ, R.C., LEE, C.S.G. *Robotics: Control, Sensing, Vision, and Intelligence.* New York, N.Y.: McGraw - Hill Book Company, 1987.
6. LASTMAN, G. J. *A shooting method for solving two-point boundary-value problems arising from non-singular bang-bang optimal control problems.* International Journal of Control, 1978, **27**( 4), pp. 513-524.
7. ARORA, J.S. *Introduction to Optimum Design.* New York, NY: McGraw - Hill Book Company, 1989.
8. IRK, D.E. *Optimal Control Theory an Introduction.* Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1970.
9. PINCH, E.R. *Optimal Control and the Calculus of Variations*. New York: Oxford University Press Inc., 1993.
10. ESCHENAUER, H., KOSKI, J., OSYCZKA, A. *Multicriteria Design Optimization*. Heidelberg, Germany: Springer-Verlag, 1990.
11. STADLER, W. *Multicriteria optimization in mechanics: A survey*. Journal of Applied Mechanics Reviews, 1984, **37**(3), pp. 227-286.
12. TIAN, L., COLLINS, C. *Motion planning for redundant manipulators using a floating point genetic algorithm.* Journal of Intelligent and Robotic Systems, Theory and Applications, **38**(3-4), pp. 297–312, 2003.
13. LEE, B.H., LEE, Y.D. *Genetic Trajectory Planner for a Manipulator with Acceleration Parametrization.* Journal of Universal Computer Science, 1997, **3**(9), pp. 1056-1073.
14. YAMAMOTO, M., MOHRI, A. *Planning of quasi-minimum time trajectories for robot manipulators (generation of a bang-bang control).* Robotica, 1989, **7**, pp. 43-47.

15. SZYSZKOWSKI, W., FOTOUHI-C, R. *A numerical method for time-optimal control of double arms robot.* IEEE Transactions on Automatic Control, 1995.
16. BOBROW, J.E., DUBOWSKY, S. *On the optimal control of robotic manipulators with actuator constraints*. Proceedings of 1983 American Control Conference.
17. *Genetic Algorithm and Direct Search Toolbox for Use with MATLAB – User's Guide*. The Mathworks Inc., Natick (USA) 2004.
18. *Optimization Toolbox for Use with MATLAB – User's Guide*. The Mathworks Inc., Natick (USA) 2000.
19. KOZÁK, S., KAJAN, S. *Matlab – Simulink I*. Bratislava: STU, 2006.
20. DISSANAYAKE, M. W, GOH, C. J., PHAN-THIEN, N. *Time-optimal trajectories for robot manipulators.* Robotica, 1991, **9**(2), pp. 131-138.
21. CHAN K. K., ZALZALA, A. M. S. *Genetic-based minimum-time trajectory planning of articulated manipulators with torque constraints.* IEE Colloquium on Genetic Algorithms for Control Systems Engineering, London, pp. 4/1 -4/3, 1993.
22. KIM, K-W, KIM, H-S, CHOI, Y-K., PARK, J-H. *Optimization of cubic polynomial joint trajectories and sliding mode controllers for robots using evolution strategy.* Proceedings of the 23rd International Conference on Industrial Electronics, Control and Instrumentation IECON 97, 1997, **3**, pp. 1444 -1447.
23. RANA, A., ZALZALA, A. *An evolutionary planner for near time-optimal collision-free motion of multi-arm robotic manipulators.* Exeter, Proceedings of International Conference on Control, pp. 29-35, 1996.
24. DOYLE, A. B., JONES, D. I. *Robot path planning with genetic algorithm.* Proceedings of 2nd Portuguese Conference on Automatic Control, 1996, pp. 312-318.
25. PACK, D., TOUSSANT, G., HAUPT, R. *Robot trajectory planning using a genetic algorithm.* SPIE, 1996, 2824, pp. 171-182.

# CONTENTS